universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

LNS-based Multi-Day Routing Heuristic for Predictive Medical Waste Collection

verfasst von | submitted by

Florian Freund BSc

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

Master of Science (MSc)

Wien | Vienna, 2026

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. Dr. Jan Fabian Ehmke
Mitbetreut von | Co-Supervisor:	Peiman Ghasemi PhD

Abstract (EN)

This master thesis addresses current challenges in medical waste collection related to inefficient routing and proposes a technology-based approach to increase cost-effectiveness while avoiding overflowing containers and considering the uncertain nature of waste accumulation. Therefore, a two-stage stochastic optimization model is proposed to solve the smart waste collection routing problem, that schedules containers based on their predicted fill levels and creates optimal routes for those containers aiming to increase operational efficiency and robustness. The demand forecasts are determined by a predictive model that considers historical and current fill levels transmitted by ultrasonic sensors attached to the inside of the container tops. The first part of this thesis focuses on solving the problem heuristically to obtain also solutions for large problem sizes within reasonable amount of time, by using a select first – route second approach entailing a Large Neighborhood Search Algorithm. In the second part of this thesis, a simulation-optimization approach is defined, where the forecasts from the predictive model are fed into a mathematical model designed based on a capacitated vehicle routing problem to obtain exact solutions. A performance evaluation by using real-world data indicated that the proposed dynamic routing approach is able to decrease the total travel distance in scenarios with lower waste accumulation and keeps the rate of overflowing containers in all tested scenarios below 5%.

Abstract (DE)

Diese Arbeit befasst sich mit der Konzipierung eines technologiebasierten Systems zur Beseitigung medizinischer Abfälle, mit dem Ziel der Optimierung der Kosteneffizienz. Dabei sollen aktuelle Herausforderungen im Hinblick auf ineffiziente Routenplanung sowie die Reduzierung überlaufender Container, unter der Berücksichtigung der Ungewissheit der täglichen Abfallproduktion, adressiert werden. Hierfür wurde ein zweistufiges, stochastisches Optimierungsmodell definiert, um das Smart Waste Collection Routing Problem (SWCRP) zu lösen, welches optimale Routen basierend auf aktuellen und zukünftigen Füllständen ermittelt, um das Überlaufen von Containern zu verhindern und gleichzeitig die operative Effektivität zu steigern und ein robustes System zur Entsorgung medizinischer Abfälle zu schaffen. Um die Containerfüllstände zu evaluieren, werden die Container mit Ultraschallsensoren auf der Innenseite des Deckels angebracht, welche in der Lage sind, den aktuellen Füllstand in Echtzeit an den Entscheidungsträger zu übermitteln. Diese Echtzeitdaten dienen im nächsten Schritt als Basis, um Prognosen für die erwarteten Containerfüllstände anhand eines Modells, unter Berücksichtigung aktueller und historischer Füllstände, zu berechnen, um potenziell überlaufende Container frühzeitig zu identifizieren. Der erste Teil dieser Arbeit zielt darauf ab, einen heuristischen Ansatz zu definieren, welcher auch für große Instanzen, Lösungen innerhalb einer angemessenen Zeitspanne ermittelt, wo hingegen exakte Methoden an Ihre Grenzen stoßen. Dafür wurde ein select first – route second Ansatz in Kombination mit einem Large Neighborhood Search Algorithmus gewählt. Der zweite Teil dieser Arbeit widmet sich einem Simulation-Optimization Ansatz, wo die ermittelten Prognosen in ein mathematisches Modell eingespeist werden, um eine exakte Lösung zu eruieren. Die Ergebnisse einer Fallstudie unter Verwendung von Daten aus der Realwelt zeigten, dass der dynamische Routing Ansatz von den Echtzeit-Füllständen profitiert und in der Lage ist, in Szenarien wo geringere Abfallmengen produziert werden, die Gesamtstrecke zu reduzieren und in Szenarien, wo höhere Abfallmengen produziert werden, die Zahl der überlaufenden Container im Vergleich zu statischen Routing-Strategien reduziert werden. Die Rate der überlaufenden Container lag in allen getesteten Szenarien unter 5%.

Table of Content

List of Figures

List of Tables

List of Abbreviations

ALNS ..... Adaptive Large Neighborhood Search

GRASP ..... Greedy Randomized Adaptive Search Procedure

IP ..... Integer Programming

IRP ..... Inventory Routing Problem

MIP ..... Mixed-Integer Programming

MILP ..... Mixed-Integer Linear Programming

MINLP ..... Mixed-Integer Non-Linear Programming

SA ..... Simulated Annealing

VRP ..... Vehicle Routing Problem

1. Introduction

1. Introduction

Waste collection plays a vital role in logistics and is continuously gaining importance as the sustainable waste management process has been recognized. The European Union for example has defined an action plan regarding a transition to a circular economy and set a common target for recycling 65% of municipal and 75% of packaging waste by 2030 as recycling can reduce landfill capacity and pollution problems. On the other hand, waste collection and recycling is very expensive, so therefore the efficiency of the waste collection system is crucial and needs to be addressed by researchers (Markov et al., 2020). Additionally, world's population is growing rapidly, especially in cities, leading to serious challenges for urban planners to design sustainable cities and waste management is one of them. To handle the increasing amount of waste in cities, efficient waste management systems must be established with low pollution and fast service. This can be achieved by utilizing real-time sensor data and transforming cities into smart cities to establish an advanced waste management system that immediately adapts to the behavior of the inhabitants (Rahmanifar et al., 2023).

Waste collection can be classified by different categories like municipal waste which includes solid household waste, recycling waste that includes items such as glass, plastic, paper and other items with common reuse materials and hazardous waste which includes materials that, if not handled properly, pose a risk to human health or the environment (Hess et al., 2024). Such hazardous waste for example can be infectious medical waste, that needs to be handled with great care as it can spread diseases and pose high risk to public health when pedestrians, health workers or waste handlers get in contact with this kind of waste like needles or syringes.

Due to the aging population, there is a fast increase of medical waste ongoing, as there are more people who get home health or medical care services. Also, global diseases lead to a tremendous increase of medical infectious waste which has created many challenges worldwide. For example, the amount of infectious waste during COVID-19 in the Hubei province, China, has increased by 370% and Spain was faced with an increase by 350% (Ghasemi et al., 2025). This is why efficient supply chain and especially medical waste management in healthcare is crucial for ensuring public health as overflowing infectious waste poses high risks to our society. Therefore, the urgent need and societal impact have driven the interest in understanding and optimizing medical waste collection to enhance operational efficiency.

Although the role of medical waste collection is critical for healthcare supply chain, organizations are faced with challenges related to monitoring and decision making since visiting a container too early leads to low service quality and unnecessary mileage and visiting a container too late results in overflowing containers and increases public health risks (Spinelli et al., 2024). This is caused by the uncertain nature of waste accumulation, as it depends on several factors and is therefore difficult to predict. In order to tackle uncertainty in medical waste collection technology can be utilized by attaching sensors to waste containers to monitor current fill levels in real time. There is already some literature existing in this field but primarily focused on municipal waste and not on infectious and recyclable medical waste. Hence, this research gap should be addressed in this thesis by proposing a novel and sustainable two-stage stochastic optimization model that addresses current challenges of medical waste collection due to uncertainty, continued aging and growth of population.

The primary objectives of this study are as follows:

1. To analyze the impact on sensors in waste containers on uncertainty in medical waste collection.
2. To propose a cost-effective dynamic routing approach for medical waste collection by using sensors in combination with a predictive model.
3. To validate the proposed approach in terms of cost-effectiveness and robustness through experiments, by comparing the results of the proposed dynamic approach with a static waste collection plan.

These objectives should provide a comprehensive understanding of how sensors can serve as an enabler to leverage cost-effectiveness of medical waste collection and how integration of technology can contribute to being better prepared for upcoming global diseases like COVID-19.

The underlying research hypothesis of this thesis is: *Integrating sensors and a predictive model in medical waste collection systems resulting in a data-driven dynamic approach to enhance cost-effectiveness*. To achieve the defined objectives and prove the hypothesis, this research applies a mixed-method approach which contains a literature review at the beginning to provide a comprehensive understanding of currently available research in the area of sensor-based medical waste management. This serves as a basis for defining a data-driven, dynamic approach that uses heuristics and metaheuristics to solve the problem and is followed by comparative

experiments with real-world data to prove cost-effectiveness and robustness of the proposed dynamic approach.

This study aims to integrate technology into the healthcare supply chain, especially in medical waste collection to enhance cost-effectiveness and improve robustness of medical waste management systems. It is important to note that the dynamic routing strategy is not solely intended for medical waste collection but can also be used in other areas. The proposed dynamic routing approach should serve as a decision support system that is able to react to increasing demands leading to being better prepared for future pandemics. This is achieved by using sensors in combination with a predictive model to monitor current fill levels and anticipate future waste accumulation rates based on historical data. Therefore, this approach also serves as an early warning system, if the increase of medical waste is exceptionally high compared to historical data, to predict local or global disease outbreaks.

By identifying critical challenges of uncertainty in medical waste collection, this research should provide insights on how the integration of technology like sensors can be used to minimize uncertainty and transform a static into a dynamic and data-driven waste collection system with enhanced cost-effectiveness by optimizing scheduling and routing decisions.

1.1. Problem Description

The uncertain nature of waste accumulation is a critical challenge for waste collection centers, as they have no access to real-time data of container fill levels. Therefore, many waste collection centers perform fixed scheduled pickups, resulting in inefficient operations (Ketzenberg & Metters, 2020). Also in this case, the waste collection center which collects infectious and non-infectious medical waste from pharmacies and hospitals is currently collecting the waste on fixed routes and time schedules. Recently, customers have expressed dissatisfaction, as vehicles often collect containers too early or too late, resulting in disturbance of the people present in the pharmacies or hospitals when visited too often or leading to exceeding containers with infectious waste which poses high risks for the people around when visited too late. Especially during months with higher infection rates, the waste collector often gets calls from their clients that their containers are already full and that they need to be emptied immediately. In such cases, the waste accumulation rate is too high and cannot be served with its available resources.

The current situation leads to customer dissatisfaction and financial losses. Thus, the waste collection center performs investigations to solve the present issues and improve customer satisfaction. As a result, they identified efficient and cost-effective waste collection as a critical challenge that must be solved. As a key to effective waste collection, minimizing uncertainty is required. A possible solution to tackle uncertainty is to attach sensors to waste containers at hospitals and pharmacies to monitor the fill levels in real-time and adapt daily route planning accordingly. This approach results in dynamic route planning, that focuses on the containers that have higher fill levels and are expected to overflow soon resulting in a reduction of unnecessary mileage of vehicles. Therefore, the integration of sensors to optimize routing and scheduling decisions will lead to a better and sustainable use of resources (de Morais et al., 2024).

As the sensor data can be evaluated continuously, the routing algorithm determines the optimal route based on current waste levels daily. The sensor also serves as an enabler to conduct demand analyses, as higher demands lead to a faster increase of waste levels. Additionally, a forecasting method will be added to the routing system to determine estimates of future waste levels based on historical data resulting in a more robust and reliable approach. Another important advantage of using a forecasting method is recognizing anomalies to adjust routings to exceptional increases in waste compared to historical data which also contributes to being better prepared for upcoming global disease outbreaks (Abaku et al., 2024).

By using sensors in combination with a forecasting approach, this thesis considers the problem of enhancing cost-effectiveness of medical infectious waste collection for closed-loop systems. In this case the decisions regarding the intervals and routes for the waste collection vehicles are made by the waste collector. Usually, the waste collection problem is tackled by using the Vehicle Routing Problem (VRP), but instead the Inventory Routing Problem is chosen, which is an extension of the VRP that also integrates inventory management and vehicle routing decisions over a medium- or long-term planning horizon. The classical IRP considers three different types of decisions that must be made. First, when to restock the customers' inventories according to their demand, secondly, how much product to deliver and lastly, how to combine customers into vehicle routing (Spinelli et al., 2024). This problem can be classified as a reverse IRP, since the reverse IRP aims to optimize vehicle routes to collect products as the inventory increases over time, whereas the traditional forward IRP aims to optimize the vehicle routes to deliver products as the inventory of customers decreases (de Morais et al., 2024).

This thesis proposes a medical waste supply chain that comprises hospitals and pharmacies, collection centers, recycling centers, and waste burners. In the first step, the medical waste is collected from hospitals and pharmacies, where health workers can drop off medical waste from their clients, and is transported to the collection centers where the non-infectious medical waste is segregated from infectious medical waste. The non-infectious waste will then be transported to recycling centers, whereas infectious waste will be transported to waste burners (see Figure 1-1).

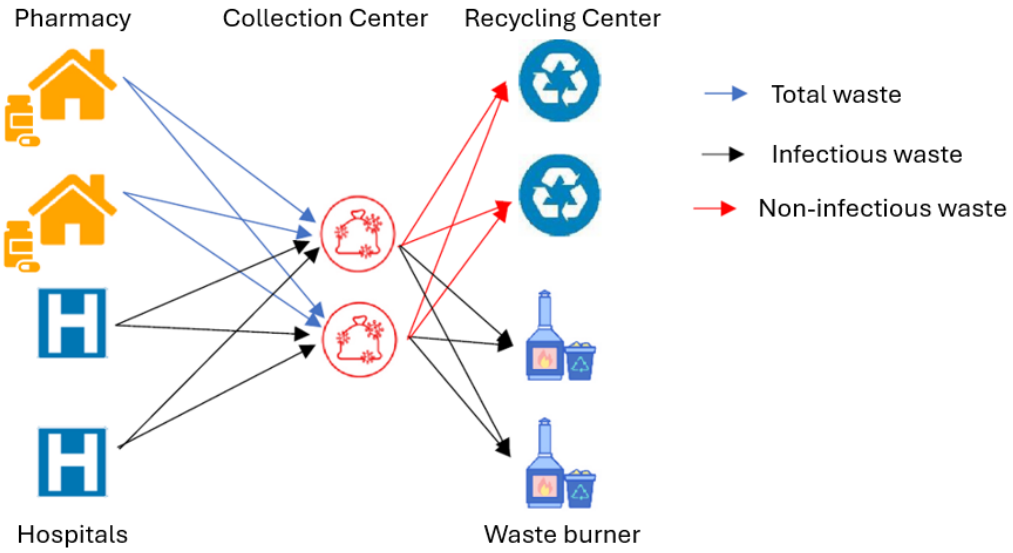

Figure 1-1 Proposed supply chain (Ghasemi et al., 2025)

The medical waste is stored in containers at pharmacies and hospitals which are emptied by a homogenous fleet of transportation vehicles, whereas each tour starts and ends at the depot. If the collection vehicle has reached its maximum capacity, it has to dispose the waste at an available dump, which are collection centers, where the infectious waste is segregated from non-infectious waste. At the end of a tour, the vehicles must visit a collection center before entering the depot (see Figure 1-2). As a consequence, we are faced with a node-routing problem, whereas municipal waste collection is an arc-routing problem (Markov et al., 2016).

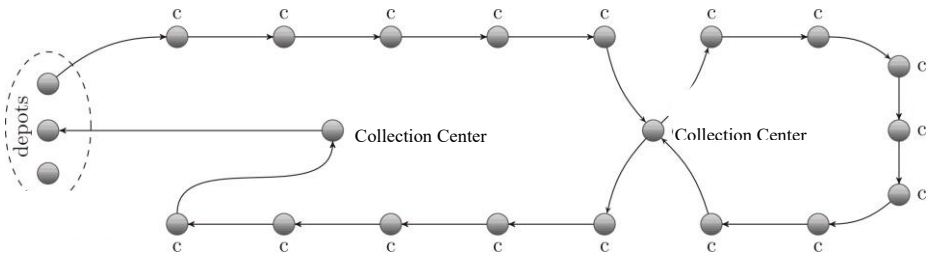

Figure 1-2 Example of a tour (Markov et al., 2016)

In this thesis, a solution should be developed to solve the dynamic and stochastic reverse IRP described above by using heuristics and meta-heuristics. The dynamic aspect ensures that the collection routes are determined daily, based on real-time sensor data, whereas static routing performs the same pre-defined routes regardless of the waste fill levels of containers. Therefore, the integration of dynamic route planning contributes to enhancing cost-effectiveness, as only those containers are visited, that are likely to overflow soon, and unnecessary mileage will be avoided.

A key consideration that the model must take into account is stochasticity, as waste rates and volumes are stochastic and fluctuate due to seasonal effects and weather conditions. For example, waste fill levels might rise faster during an influenza epidemic, such that containers must be visited more often. So even if sensors are used, it makes sense to assume demands to be stochastic, since epidemics are hardly predictable (Hess et al., 2024). In this study, the sensors should be attached to the inside of the container lid to measure the current fill level and simultaneously reduce uncertainty associated with bin fill levels. Uncertainty plays a vital role in waste collection, as unknown fill levels lead to routes that visit empty bins or bins overflowing due to lack of visits resulting in poor cost-effectiveness (de Morais et al., 2024). The real-time data is then transmitted to the decision maker daily and serves as a basis for dynamic route planning. But the research for municipal waste has also shown that attaching sensors can be cost intensive, as all containers have to be equipped with technology. In the scenario of medical waste collection, only a limited number of hospitals and pharmacies have to be visited and equipped with sensors, resulting in lower costs and therefore hardly impacting cost-effectiveness. This hypothesis must be proven by experiments which will be performed after developing an approach to solve the dynamic reverse IRP and the results should then be compared to a fixed routing schedule without sensors, so that the performance of the proposed approach can be determined.

To solve the described problem, a two-stage stochastic optimization model should be formulated, following a select first – route second approach. In the first stage, current fill levels are being analyzed and if containers exceed a specified threshold level, they are scheduled for collection on that day. In the second stage, initial routes are determined by a Nearest Neighbor algorithm for each day of the planning period and optimized by applying a Large Neighborhood Search (LNS) to merge routes resulting in decreased total travel distance. This will result in a sequential approach where heuristics, metaheuristics, and a predictive model are combined to

obtain a solution for the dynamic and stochastic reverse IRP. Several sources of evidence support the identification and analysis of this solution in order to increase the cost-effectiveness of waste collection by integrating sensors.

1.2. Research Questions

This thesis aims to enhance the cost-effectiveness and robustness of medical waste collection by developing a two-stage stochastic optimization model to solve the reverse IRP. In order to develop a dynamic approach that utilizes real-time sensor data to optimize container scheduling and route planning, research questions are defined to guide the process of development in the right direction.

Main Research Question:

1. How to enhance cost-effectiveness of medical waste collection by using real-time data provided by sensors?

Secondary Research Questions:

1. What are the key cost drivers affecting the cost-effectiveness of medical waste collection?
    - This question explores the variables that influence waste accumulation and the related demand for medical products.
2. What role does uncertainty play in enhancing cost-effectiveness of medical waste collection?
    - This question focuses on the aspect of uncertainty and how it affects effective medical waste collection.
3. How can real-time data of sensors be leveraged to transform static into a dynamic route planning for medical waste collection to improve operational effectiveness?
    - This question aims to identify an innovative approach to improve medical waste collection in terms of customer satisfaction and cost-effectiveness.
4. How can the integration of a predictive model be leveraged to improve robustness of a sensor-based medical waste management system?
    - This question focuses on understanding the role of integrating a predictive model to forecast future demands based on historical data in enhancing robustness.

1.3. Research Objectives

In the next step, research objectives are determined ensuring that the development and implementation process is guided to answer the previously defined research questions:

1. Face with uncertainty of waste accumulation by using sensors on medical waste containers which serve as an enabler to monitor current fill levels.
    - This objective focuses on creating a technology-driven approach that reduces uncertainty to optimize medical waste collection in terms of cost-effectiveness and more sustainable use of resources.
2. To develop a sequential approach that considers current waste fill levels, to solve the reverse IRP for medical waste collection by using heuristics.
    - This objective entails creating a sequential approach, select first – route second, that integrates sensors in waste containers to solve the reverse IRP and enabling dynamic route planning for the medical waste collector.
3. To enhance cost-effectiveness of medical waste collection by transforming fixed into dynamic route planning.
    - This objective aims to demonstrate the advantages of using dynamic instead of fixed route planning in medical waste collection.
4. To integrate a predictive model that enables better estimations of future waste accumulation to improve robustness of the proposed approach.
    - This objective entails creating a comprehensive approach that integrates a predictive model that enables predicting future demands resulting in improved route planning.
5. To develop a data-driven dynamic waste collection approach that contributes to detecting increased demands resulting in being better prepared for upcoming local and global diseases.
    - This objective aims to create a novel approach that is able to detect extraordinary increases in waste accumulation compared to historical data and therefore contributes to being better prepared for future pandemics.
6. To evaluate the cost-effectiveness of dynamic route planning compared to the static approach through experiments.
    - This objective emphasizes empirical testing to confirm the approach's cost-effectiveness compared to fixed routing.

1.4. Research Contributions

This research aims to make numerous key contributions to medical waste management by proposing a dynamic routing approach by utilizing real-time sensor data in combination with heuristics and metaheuristics:

- **Novel heuristic approach:** This study develops a novel heuristic approach using a Large Neighborhood Search and a predictive model to solve the reverse IRP under uncertainty. This enables obtaining feasible solutions also for larger instances, especially when exact solvers experience their computational limits and solutions cannot be obtained within a reasonable amount of time. Furthermore, the predictive model contributes to reducing uncertainty of waste accumulation, since future fill levels are anticipated based on historical data.
- **Comparative analysis with traditional static routing strategy:** A detailed analysis of the proposed dynamic approach compared to a traditional static strategy is performed under several circumstances to examine the behavior in terms of cost-effectiveness and robustness in different scenarios. This provides insights for which scenarios a dynamic approach is most suitable.
- **Robustness and flexibility:** To address the challenges of medical waste collection, robustness of the proposed dynamic routing strategy is studied aiming to avoid containers from overflowing and consequently reducing public health risks. This also contributes to designing a flexible approach which can adapt to changing waste accumulation environments and therefore provides a practical decision support tool.

1.5. Structure of the Thesis

The thesis is structured to provide a detailed analysis of medical waste collection, its challenges and how to tackle them by solving the reverse IRP under uncertainty by utilizing real-time sensor data in combination with a Large Neighborhood Search.

- **Chapter 2: Literature Review**
  This chapter presents a comprehensive review of existing literature starting with the Inventory Routing Problem, followed by Sensor-based waste management and closing with medical waste management. This aims to gradually provide an in-depth overview

of the current state of research in sensor-based medical waste management. Based on this, research gaps are identified which should be addressed by this thesis.

- **Chapter 3: Methodology**

  This chapter describes the mathematical and heuristic approach to solve the reverse IRP under uncertainty for medical waste collection. Since this thesis aims to provide a practical approach, it provides an in-depth description of the implementation of the proposed two-stage stochastic optimization model starting with the predictive model in phase one, followed by the Large Neighborhood Search algorithm in phase two.

- **Chapter 4: Data Analysis and Results Discussion**

  This chapter demonstrates the comparative evaluation of static and dynamic routing strategies. First, the real-world data used for experiments is described, followed by an analysis of the arrival rates and the accuracy of the predictive model using real-world data. Afterwards, the performance analysis of each routing strategy is performed to examine the cost-effectiveness and robustness aiming to answer the research questions.

- **Chapter 5: Conclusion**

  This chapter summarizes the key findings of the data analysis and simulations focusing on answering the research questions. Furthermore, it outlines the limitations of this thesis and provides an outlook on future work.

2. Literature Review and Research Gaps

2.1. Introduction to Literature Review

This section aims to present an overview of present literature related to sensor-based medical waste collection. Based on this, current research gaps and challenges should be identified and addressed by this work. The literature Review is structured to gradually provide an in-depth overview of sensor-based waste management for medical waste by starting with the Inventory Routing Problem, which is well suited to describe waste collection procedures since it also considers inventory additionally to routing decisions. The overview of IRPs is followed by a review of literature on sensor-based waste management to present an overview of current research in this area. In the next step, research in medical waste management is examined to complete the literature review on sensor-based waste management for medical waste. Based on the findings from literature review, research gaps are identified that will be addressed by this thesis.

2.2. Overview of the Inventory Routing Problem (IRP)

The IRP was initially introduced by Bell et al. (1983) as decision support system for air products and chemicals. It integrates customer demand forecasting, a shortest path algorithm, and a mathematical optimization module to determine daily delivery schedules. The optimization module contains a Lagrangian relaxation algorithm to solve mixed integer programming, was first implemented in October 1981 and resulted in savings of operating costs between 6% and 10%.

Trudeau & Dror (1992) studied a stochastic IRP for the distribution of a heating oil. The customers' consumption is assumed to be stochastic to address the uncertain nature of demands since demand depends on seasonal effects which are hard to predict. The authors performed a detailed analysis of the stochastic problem including the aspect of route failures when actual demand exceeds the capacity of a vehicle. The best results achieved the strategy where customers are assigned heuristically, followed by solving the VRP using a Clark and Wright algorithm which was tested on real-world data from 12 weeks.

Bard et al. (1998) presented a decomposition approach for solving the IRP under uncertainty, since customer demands are not known with certainty. The proposed approach is developed by integrating a balanced assignment problem to allocate customers to specific days, followed by three heuristics – randomized Clark and Wright, GRASP, and modified Sweep algorithm – to solve the routing problem. To verify the performance of the proposed approach, tests on datasets from a liquid propane distributor were performed.

Also Campbell and Savelsbergh (2004) proposed a decomposition approach where a delivery schedule is created in the first step using integer programming, followed by the construction of delivery routes utilizing routing and scheduling heuristics. This two-phase approach is applied on a rolling horizon framework and aims to determine a distribution strategy that minimizes long-term distribution costs demonstrated by performing computational experiments.

Coelho et al. (2014) analyzed the performance of numerous heuristics to solve the dynamic version of the IRP where customer demands are uncertain and forecasts are made for planning. The authors performed extensive computational experiments to examine the performance of each heuristic on randomly generated instances. The key findings showed that longer rolling horizon periods do not improve solutions and that using stochastic information improves solution quality but requires much more computational time.

Alarcon Ortega and Doerner (2023) presented a two-stage mathematical program to solve the stochastic IRP, since customers' demands are uncertain. They developed a matheuristic solution approach based on an Adaptive Large Neighborhood Search algorithm. To examine the models' performance, the results are compared with the branch and regret algorithm from literature. The results show that the proposed model outperformed other methods by over 40%.

Charaf et al. (2024) developed a two-echelon IRP aiming to minimize total costs by optimizing routing decisions. To solve the problem, a two-phase matheuristic was presented which combines tabu search and mathematical programming. Experiments verified the performance of the proposed approach, which showed excellent solution quality.

Feng et al. (2024) presented a framework to address the uncertain demand by developing a distributionally robust IRP model by utilizing the worst-case mean-conditional value-at-risk criterion. A case study demonstrated that the proposed model provides a robust delivery schedule and simultaneously reduces the product shortage rate.

Gunawan et al. (2024) developed a distribution model of industrial gases aiming to maximize profitability. The model utilizes the Visit Frequency Optimization Procedure, Nearest Neighbor, 2-opt Algorithm, and Taboo Search to establish an efficient model for distribution. Efficiency of the proposed approach was verified with experiments showing increased profitability.

Alarcon Ortega et al. (2024) developed an iterative lookahead algorithm to solve the stochastic IRP aiming to support suppliers by creating efficient replenishment plans. The proposed approach utilizes Adaptive Large Neighborhood Search to determine optimal routes and policy

learning for replenishment decisions. The effectiveness of the iterative lookahead algorithm was verified by performing numerical evaluations.

Zheng et al. (2024) presented a solution for the IRP under uncertainty for an incentive-based recycling system. The problem was solved by developing a three-phase iterative algorithm by combining the progressive hedging algorithm and route splitting algorithm based on the Lin-Kernighan heuristic. To examine the effectiveness of the proposed algorithm, a case study was performed showing a decrease of total costs by up to 42% compared to a Genetic Algorithm.

Haseltalab & Karimi (2025) proposed two solutions to heuristically solve the deteriorating IRP by utilizing a Genetic Algorithm and Tabu Search (GATS) and a Genetic Algorithm and Greedy sub-algorithms (GAG). To evaluate the performance of the proposed algorithms, tests were performed followed by a comparative analysis showing that the Genetic Algorithm Tabu Search produced optimal solutions, whereas the other solution led to satisfactory results.

Fu et al. (2025) proposed a hybrid method to optimize the vehicle scheduling problem in port cargo transportation based on a combination of an Ant Colony Optimization and a Genetic Algorithm. Experiments verified the effectiveness of the proposed approach in improving transportation efficiency.

Yu et al. (2025) presented the Blood Stochastic IRP aiming to reduce total costs while maintaining service level quality. To solve this problem, a three-stage matheuristic was developed that combines a perturbation heuristic, Adaptive Large Neighborhood Search, and an exact approach. Experimental simulations with real-world data prove the effectiveness of the proposed approach compared to a two-stage matheuristic.

Sensor-based Waste Management

Mes et al. (2014) presented a dynamic approach to solve the reverse IRP under uncertainty. The uncertain nature is tackled by using an anticipatory policy. The heuristic classifies containers based on their fill levels into MustGo's, MayGo's, and NoGo's. Based on their category, the containers are added to routes or not. The approach is then tested on real-world data provided by a waste collection company from the Netherlands, indicating cost savings up to 40%.

Elbek et al. (2015) addressed the problem of collecting recyclable waste under uncertainty, since waste accumulation is assumed to be stochastic. The authors aim to minimize operational costs while avoiding violations of capacity constraints from vehicles and containers. To solve this problem a two-stage stochastic programming formulation with single- and multi-period

policies was developed. Computational experiments showed that multi-period policy outperformed the single-period policy and both approaches benefit from sensor data.

Markov et al. (2020) tackled the stochastic IRP for collecting recyclable waste from containers equipped with sensors measuring the current fill levels. The sensor data is then used to anticipate future fill levels by using a forecasting model. To solve this problem an ALNS algorithm with an integrated predictive model was developed and tested on real-world data from a Swiss waste collection company from Geneva. Results showed that the proposed approach outperforms deterministic policies in terms of overflowing containers and operation costs.

Cubillos et al. (2022) studied the stochastic IRP by using sensor equipped containers to derive delivery decisions aiming to reduce total costs. Since number of sensors is limited, numerous assignment rules are examined to find the most promising rule in terms of total costs. To evaluate the rules, a Variable Neighborhood Search in a rolling horizon framework is implemented. Results showed that simple rules like placing sensors at customers with highest demand lead to promising reductions of total costs.

Jorge et al. (2022) tackled the smart waste collection problem by developing a hybrid metaheuristic. In the first step, future fill levels are anticipated by a look-ahead heuristic to determine must-go containers which must be visited for collection. In the second step, a simulated annealing / neighborhood search algorithm is performed to determine most profitable routes. The proposed approach is tested using an instance of up to 500 containers. A real-world case study showed that the profit from recyclables can be increased by 45% compared to the traditional approach.

Tandon & Bansal (2023) examined IoT-based waste management by using sensors to monitor current fill levels aiming to improve the efficiency of garbage collection. The sensor data is used to detect overflowing bins which are communicated to waste collectors enabling them to take corrective actions. Additionally, the sensor data is used to optimize collection routes resulting in an efficient waste management system.

De Morais et al. (2024) analyzed inefficiencies in waste collection due to high uncertainty of bin fill levels. To tackle this problem the authors developed a data-driven optimization approach to solve the dynamic reverse IRP. The proposed model utilizes machine learning to recognize patterns and anticipate future fill levels to reduce uncertainty. To demonstrate the practicability of such an approach, a real-world case study was performed that showed a significant improvement in operational efficiency.

Hussain et al. (2024) proposed a multiagent simulation-based framework to assess the dynamics of an IoT-based waste management system. The system utilizes sensors to measure waste production which is then used by a predictive routing system to determine the most efficient routes. To demonstrate the performance, a comparative analysis with a traditional review strategy was performed, showing that the proposed system outperformed a traditional strategy.
Marković et al. (2024) presented a prototype for smart waste collection by solving a dynamic VRP by using heuristics aiming to optimize routes and reduce total costs. The prototype analyzes real-time sensor data from containers which are scheduled for collection if fill level exceeds 75% and a Clark and Wright Savings Algorithm is used to solve the dynamic VRP.
Saraswathi P et al. (2024) proposed an IoT-based waste management system that utilizes sensors to segregate waste and monitor current fill levels. The improved effectiveness in segregating waste allows to enhance environmental health while monitoring current fill levels leads to an increased effectiveness of resource und route planning.
T et al. (2024) addresses an IoT-driven food waste monitoring system gathering information of real-time waste data aiming to develop an effective and sustainable waste management system. This is ensured by attaching sensors to waste containers resulting in smart bins. To demonstrate the advantages of using smart bins, a case study was performed.
Thangam et al. (2024) proposed a smart waste management system by using IoT sensors to enhance collection efficiency and sustainability. Therefore, ultrasonic sensors are attached to waste containers to monitor current fill levels allowing waste collectors to dynamically adjust collection routes accordingly.
Samyudha K et al. (2025) addressed the lack of real-time monitoring by using IoT-based smart bins to optimize waste management. The ultrasonic sensor in the waste bin monitors current fill levels and alerts waste collectors if the bin is full contributing to enable seamless waste management and reducing manual monitoring resulting in a more sustainable waste management system.
SureshKumar et al. (2025) proposed a smart bin system to enhance effective waste management. The approach utilizes sensors in waste containers to monitor current fill levels and send alerts to waste collectors resulting in optimized resource allocation and routing decisions.
Swapna et al. (2025) demonstrated an IoT-driven intelligent waste management system utilizing sensors to enable real-time monitoring, dynamic route optimization, and automated disinfectant mechanisms aiming to improve public health and operational efficiency. Therefore,

sensors are attached to waste containers enabling to dynamically adapt collection routes to current fill levels by using a modified Travelling Salesman Problem algorithm.

2.3. Medical Waste Management

Nolz et al. (2014b) considered the stochastic IRP for infectious medical waste collection aiming to minimize public health risks. The waste containers at local pharmacies are equipped with sensors measuring current fill levels. To solve this problem, an approach based on an ALNS algorithm was developed. To evaluate the performance of the proposed model, tests on real-world data were performed showing that the ALNS based approach leads to good approximations compared to optimal solutions from exact solver.

In Nolz et al. (2014a) studied the stochastic IRP for infectious medical waste collection from pharmacies, disposed by home-health and medical care services, aiming to minimize total costs while increasing service level quality. To solve this problem, three heuristic approaches based on a LNS algorithm are proposed to solve the bi-objective stochastic IRP and extensively tested on real-world datasets.

Osaba et al. (2019) presented a solution for a real-world drug distribution problem with pharmacological waste collection by solving a rich vehicle routing problem (RVRP). To solve the RVRP, a Discrete and Improved Bat Algorithm was developed. To examine the performance of the novel approach, comparisons with three other approaches were performed, consisting of an evolutionary algorithm, an evolutionary simulated annealing and a firefly algorithm. The key findings demonstrate that the proposed approach delivers promising results for addressing the RVRP.

Taslimi et al. (2020) addressed the Periodic Load-dependent Capacitated Vehicle Routing Problem (PLCVRP) for medical waste collection companies to design an inventory routing schedule on a weekly basis aiming to minimize transport and occupational risks. To solve this problem, a decomposition-based heuristic is developed. To demonstrate the performance of the proposed approach, a real-world case study was performed.

Wu et al. (2020) presented a priority considered green VRP for waste management systems that specifically focuses on collecting medical waste which is classified as high-priority waste. To solve the problem, a hybrid algorithm is developed consisting of a combination of a Particle Swarm Optimization and a Simulated Annealing algorithm. Experimental tests showed that the proposed approach led to a decrease in 42,3% in environmental impact compared to traditional approaches.

Govindan et al. (2022) developed a multi-period, and bi-objective mixed-integer linear-programming model for medical waste management under uncertainty aiming to minimize total costs and population risk. A key innovation was the use of queuing theory to manage truck waiting times at treatment centers. To solve the bi-objective MILP an augmented epsilon-constraint method was applied, with its effectiveness demonstrated through a case study.

Divakaran et al. (2023) presented an IoT-based smart bin system that utilizes sensors to detecting dangerous gases and avoid container overflows aiming to minimize the potential for transmission of infectious diseases for healthcare workers. The proposed system enables real-time monitoring which allows collectors to react to changing environments and update collection routes immediately resulting in more efficient waste management system.

Ishaq et al. (2023) studied on how to implement Internet of Things to improve effectiveness on biomedical waste management. Therefore, sensors were used to monitor container fill levels to ensure timely collection. The study included a set of ten hospitals and demonstrated the significance of IoT in efficient waste management.

Pulparambil et al. (2024) presented a smart medical waste management system providing real-time monitoring on current fill levels. If a container exceeds its fill level, an alert is sent to stakeholders so that they can take corrective actions. Experiments showed that the system works efficiently without the need for manual checking and allows healthcare institutions to ensure transparency and environmental cleanliness.

Goodarzian et al. (2024) proposed a Fuzzy Inference System to forecast medical waste accumulation aiming to provide a model that enables detoxification centers to make reliable forecasts on medical waste generation. The proposed model aims to minimize costs, reduce environmental impact, and establish detoxification centers. To validate the models' performance, a real case study to evaluate practicability.

Mestouri et al. (2024) examined the challenges of hazardous waste management by utilizing a combination of geographic information systems and Ant Colony Optimization to solve the Capacitated VRP. A case study in Tunisia including 174 hospitals demonstrated the performance of the proposed model, indicating reduced total travel distance and improved vehicle efficiency.

Rajakumari S T & Roy (2024) presented a smart waste management system that utilizes IoT devices aiming that waste bins are only visited when required resulting in minimized total costs. Furthermore, the system uses sensors to identify hazardous gases and transmits the data to a

central monitoring system so that the waste collector can take corrective actions resulting in a prioritized approach that increases cost-effectiveness and maintains public health.

Ghasemi et al. (2025) proposed a multi-objective, multi-echelon, and multi-commodity model to address municipal medical waste management aiming to minimize costs and reduce the risk of outbreaks. The problem is solved by using the Non-dominated Sorting Genetic Algorithm II. To evaluate the performance of the proposed approach, it was benchmarked with an exact solver and validated through a real case study.

Zhang (2025) presented a variable cycle-based medical waste recycling network optimization method to tackle the lacking efficiency of traditional fixed cycle strategies. To solve the problem a Genetic Algorithm is used to adjust the routes dynamically aiming to reduce total costs. A case study with 30 healthcare facilities demonstrated that the proposed approach outperforms the traditional fixed cycle strategy.

Table 2-1 A review of literature

Author	Variant of VRP	Method	Objective	Solution technique	Type
Bell et al. (1983)	IRP	MIP	Minimize cost	Lagrangian relaxation algorithm	Industrial gases
Trudeau & Dror (1992)	Stochastic IRP	Clarke and Wright Algorithm	Minimize cost	Heuristic (Best replenishment day + Generalized Assignment Problem + Clarke and Wright + k-opt)	Heating oil
Bard et al. (1998)	IRP	MILP	Minimize cost	Heuristics (randomized Clarke-Wright + GRASP + modified sweep)	Commodity
Campbell & Savelsbergh (2004)	IRP	IP	Large-scale applicability	Heuristics (Insertion / GRASP)	Industry gases
Coelho et al. (2014)	Dynamic and stochastic IRP	MILP	Minimize cost	Heuristic (ALNS)	General
Mes et al. (2014)	Stochastic Dynamic IRP	MILP	Minimize cost	Heuristic (Problem-based)	Solid waste
Nolz et al. (2014a)	Stochastic IRP	Two-stage stochastic optimization problem	Minimize cost and health risk	Heuristic (ALNS)	Medical waste
Nolz et al. (2014b)	Stochastic IRP	MILP	Minimize cost	Heuristic (ALNS)	Medical products
Elbek et al. (2015)	IRP	Two-stage stochastic programming	Minimize cost	Heuristics (single- and multi-period policies with and without using sensors)	Recyclable Materials

Osaba et al. (2019)	Rich VRP	Bat Algorithm (Metaheuristic)	Minimize cost	Metaheuristic (Discrete and Improved Bat Algorithm)	Pharmacological waste
Markov et al. (2020)	Stochastic IRP	MINLP	Minimize cost	Heuristic (ALNS+SA+RH)	Solid waste
Taslimi et al. (2020)	Periodic Load-dependent Capacitated VRP	ILP	Minimize transport and occupational risk	Decomposition-based heuristic	Medical waste
Wu et al. (2020)	Priority Considered Green VRP	Local Search Hybrid Algorithm	Minimize greenhouse gas and costs	Heuristics (Particle Swarm Optimization + SA	Medical waste
Cubillos et al. (2022)	Stochastic IRP	Variable Neighborhood Search	Optimal sensor allocation	Heuristic (Variable Neighborhood Search) within a rolling horizon framework	General
Jorge et al. (2022)	Smart Waste Collection Routing Problem	Simulated Annealing / Neighborhood Search	Minimize computing time and costs	Heuristics (Look-Ahead + SA / Neighborhood Search)	General
Govindan et al. (2022)	Location-routing Problem	MILP	Minimize total cost and population risk	Augmented Epsilon Constraint Method	Medical Waste
Alarcon Ortega & Doerner (2023)	Stochastic IRP	Two-stage mathematical Problem	Minimize cost	Metaheuristic (ALNS)	General
Ishaq et al. (2023)	IRP	Case Study	Implementation of Internet of Things sensors to enhance efficiency	Experiments	Biomedical waste

Alarcon Ortega et al. (2024)	Stochastic IRP	Finite-horizon stochastic dynamic program	Minimize total costs	Iterative lookahead algorithm with an ALNS and policy learning	Groceries
Charaf et al. (2024)	IRP	Matheuristic	Minimize total costs	Tabu Search and mathematical programming models	General
de Morais et al. (2024)	Dynamic IRP	Mathematical Model	Minimize uncertainty and cost	Exact solution method using mathematical model	General
Feng et al. (2024)	Stochastic IRP	Distributionally robust M-CvaR IRP model	Minimize uncertainty and optimize routing	Incorporate the worst-case mean-conditional value-at-risk (M-CvaR) criterion	General
Goodarzian et al. (2024)	Sustainable COVID-19 Medical Waste Supply Chain Problem	Grasshopper Optimization / Tabu Search	Forecast medical waste	Heuristic (Modified Grasshopper Optimization Algorithm)	Medical waste
Gunawan et al. (2024)	Flexible Periodic VRP	Heuristics	Maximize profitability	Visit Frequency Optimization Procedure, Nearest Neighbor, 2-opt Swap, and Tabu Search	Gas cylinders distribution
Hussain et al. (2024)	Multiple TSP	Mathematical Model	Maximize cost-efficiency	Integer Linear Program	Municipal waste
Marković et al. (2024)	Dynamic Capacitated VRP	Heuristics	Maximize efficiency	Clark and Wright Savings Algorithm	Municipal waste
Mestouri et al. (2024)	Capacitated VRP	Heuristics	Maximize efficiency	Geographic Information Systems with Ant Colony Optimization Algorithm	Hazardous medical waste

Zheng et al. (2024)	Recyclable IRP	Three-phase iterative algorithm	Minimize loss of recyclables and logistics costs	Progressive hedging algorithm and route splitting algorithm based on the Lin-Kernighan heuristic	Recyclable waste
Fu et al. (2025)	VRP	Hybrid optimization algorithm	Maximize efficiency	Ant Colony Optimization Algorithm and Genetic Algorithm	Port logistics
Ghasemi et al. (2025)	Medical Waste Supply Chain Problem	Multi-echelon mathematical model	Minimize cost and risk of COVID-19 outbreak	Metaheuristic (Non-dominated Sorting Genetic Algorithm II)	Medical waste
Haseltalab & Karimi (2025)	Deteriorating IRP	Mixed Integer Non-Linear Programming model	Optimize routing costs, minimize evaporation losses, and optimize inventory levels	Genetic Algorithm and Tabu Search (GATS) Genetic Algorithm and Greedy sub-algorithms (GAG)	Petrol station replenishment
Yu et al. (2025)	Stochastic IRP	Three-stage matheuristic	Maximize efficiency	Perturbation heuristic, ALNS, and an exact approach.	Blood distribution
Zhang (2025)	Medical Waste Recycling Problem	Variable cycle-based medical waste recycling network optimization method	Maximize efficiency	Multi-period, multi-vehicle mixed-integer planning model combined with a Genetic Algorithm.	Recyclable medical waste
This study	Dynamic and stochastic reverse IRP	Two-stage stochastic optimization problem	Minimize cost	Heuristics	Medical waste

2.4. Research Gaps

Based on the literature review, numerous research gaps have been identified:

- Lack of attention to integration of real-time data for medical waste collection.
  Most existing studies neglect the uncertain nature of waste accumulation resulting in less accurate and less cost-effective approaches. Therefore, research on stochastic approaches is required to leverage real-time sensor data to maximize forecasting accuracy and consequently enhancing cost-effectiveness and service level quality.
- Lack of attention to overflow prevention and health risk reduction by using sensors.
  Overflowing containers of medical waste poses enormous health risks to society, as it allows diseases to spread rapidly. However, the prevention of overflowing containers has not been adequately addressed in scientific research so far, since it can contribute to being better prepared for future pandemics.
- Absence of dynamic route planning in medical waste collection.
  There is a lack of research on dynamic route planning based on real-time sensor data to optimize cost-effectiveness in medical waste collection. Additionally, dynamic route planning allows to immediately adapt to changing environments and therefore avoids overflowing containers to maximize service level quality and reduce health risks
- Insufficient analysis of the benefits of dynamic vs. fixed route planning.
  Since static routing strategies are widely used because of lacking real-time data, there is limited research on comparative analysis between static and dynamic routing strategies. Therefore, a detailed evaluation of dynamic routing strategies in terms of cost-effectiveness and robustness compared to traditional approach is required.
- Underutilization of sensors for predicting future waste fill levels.
  There is a lack of research on the potential of reducing uncertainty and increasing service level quality by integrating a predictive model into medical waste management systems. Furthermore, a predictive model can also contribute to identifying upcoming epidemics and consequently provides an early warning system to being better prepared for future pandemics.

Addressing these research gaps plays a vital role in improving the cost-effectiveness and robustness of medical waste collection to ensure sustainable and resilient health management.

2.5. Addressing Identified Research Gaps

The literature review has shown that there are numerous research gaps existing in medical waste collection, especially in terms of using real-time sensor data to monitor current fill levels and leveraging that data in combination with historical data to anticipate future fill levels and consequently reducing uncertainty of waste accumulation. This provides potential for optimized scheduling and routing decisions, resulting in increased cost-effectiveness and decreased container overflows due to reduced uncertainty.

Therefore, this study aims to propose a mathematical model and a heuristic approach, especially tailored for medical waste management, to tackle cost-effectiveness while minimizing uncertainty and container overflows. To heuristically solve the problem, a two-stage stochastic optimization model is developed by combining a predictive model with a Large Neighborhood Search algorithm, whereas the predictive model addresses uncertainty and the Large Neighborhood Search algorithm ensure optimized routing decisions. To solve the problem exactly, the forecasts from the predictive model are fed into a mathematical model aiming to obtain optimal routes based on predicted fill levels. Afterwards, a comparative analysis based on real-world data is performed to evaluate the performance of the proposed dynamic approach compared to a traditional routing strategy to provide insights whether sensor-based medical waste management results in increased cost-effectiveness and robustness.

The identified research gaps indicate the importance of developing novel dynamic approaches for medical waste management to tackle current challenges related to uncertainty, cost-effectiveness and service level quality. By implementing a dynamic approach that utilizes real-time sensor data and a predictive model to reduce uncertainty, this study contributes to address present challenges in medical waste collection aiming to provide an efficient and resilient waste management system.

3. Methodology

3.1. Introduction

This chapter provides a detailed description of the methodologies applied to solve the stochastic and dynamic reverse IRP. This approach entails a forecasting model to anticipate future waste accumulation based on a normal distribution, to address the stochastic nature of waste accumulation. Based on these forecasts, the dynamic routes are planned daily, resulting in a stochastic and dynamic approach that only visit containers when necessary, aiming to increase cost-effectiveness. In small instances, this problem can be solved using exact methods such as Mixed-Integer Linear Programming (MILP). However, due to the combinatorial explosion of route and time assignment possibilities in real-world settings, exact approaches become computationally intractable for larger instances. Consequently, this thesis also adopts a metaheuristic approach, specifically a Large Neighborhood Search-based multi-day routing heuristic, to construct and iteratively improve collection routes of a planning period. This method balances solution quality and computational efficiency by combining Large Neighborhood Search (LNS) strategies that reassign containers across days to increase cost-effectiveness.

The proposed MILP and the LNS-based Multi-Day Routing Heuristic provide a basis to tackle the challenges of the stochastic and dynamic reverse IRP and are presented in this chapter in detail, which allow to obtain exact solutions for small instances and near-optimal solutions for larger instances.

3.2. Mathematical Formulation

This section describes the mathematical formulation of the stochastic and dynamic reverse IRP for small instances, which combines two critical components: determining when to visit a container based on an expected accumulation rate and routing decisions. Both components must be optimized to ensure maximum cost-effectiveness. At the same time overflowing containers must be avoided, as medical waste is infectious and poses high risk to society. The problem is modeled as a Smart Waste Collection Routing Problem based on existing literature, where routes are determined daily based on current container fill levels, while considering the uncertainty of customer demand.

3.2.1. Problem Definition

The mathematical formulation for this problem was proposed by Ramos et al. (2018) to solve the Smart Waste Collection Routing Problem (SWCRP). It was defined by a given set of n waste containers, k identical vehicles with a limited capacity Q in kilograms (kg), and a depot where all vehicles begin and end their routes, a complete undirected graph with n nodes is constructed whereas each node represents a waste container i with a maximum capacity E_i. The n nodes are connected by an edge (i, j) with distance d_{ij}. Additionally, a fixed cost C is incurred per unit of distance travelled. The representation of the SWCRP is shown in Figure 3-1.

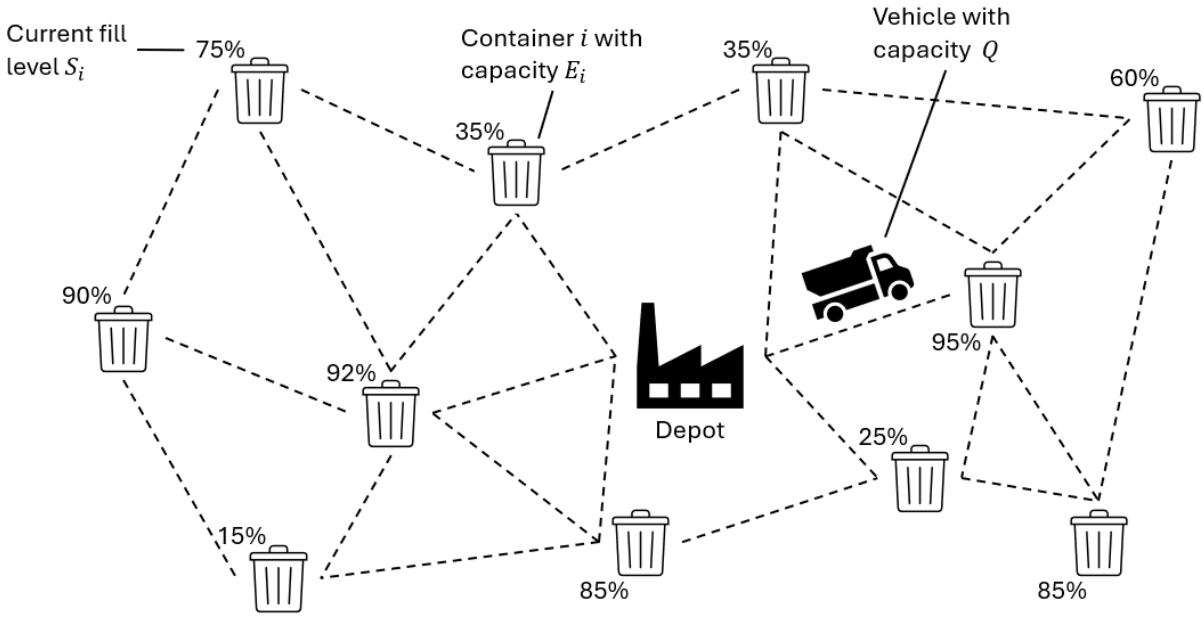

Figure 3-1 Representation of the SWCRP

On a daily basis, the sensors attached to the top inside the container transmit their current fill level in percent to the monitoring system. Therefore, the system maintains the current fill level of each container S_i. Based on historical data, an average arrival rate for each container a_i is stochastically determined resulting in an expected fill level F_i. Based on this information, the model schedules the containers that are expected to exceed the threshold ψ before the next possible collection and determines optimal routes aiming to minimize total travel distance, as shown in the flowchart in Figure 3-2.

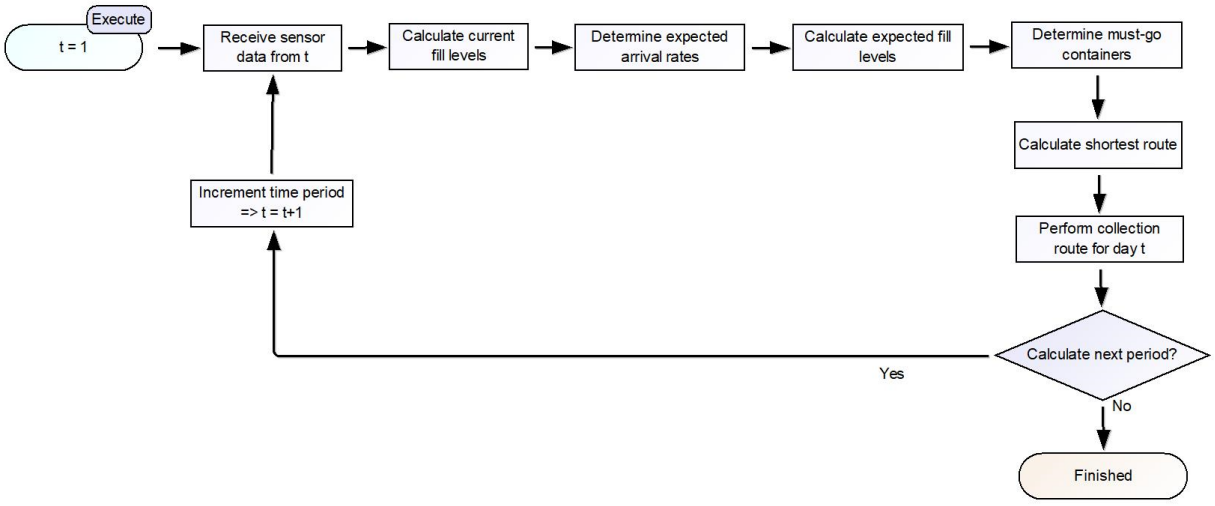

Figure 3-2 Flowchart oft he SWCRP diagram

The objective is to determine which containers to visit and the optimal route for the vehicle on each day t, in order to minimize total travel distance, while respecting vehicle capacity, container capacity, and container fill levels.

Key Assumptions

- **Capacity constraints:** Vehicles and containers have capacity constraints that must be considered during decision making.
- **Stochastic demand:** Since medical waste accumulation is hard to predict due to numerous factors like weather conditions and seasonal effects, the expected arrival rates are stochastic and follow a normal distribution which is considered by a forecasting model.

Sets and Indices

- $I = \{0, 1, 2, \dots, n\}$: Set of n waste containers starting with the real depot 0 and ends with container n.
- $i, j \in I$: Indices for waste containers

Parameters

- C: Cost of travelling per distance unit in Euro (€).
- Q: Vehicle capacity in kg.
- d_{ij}: Distance between node i and node j.

- S_i: Amount of waste in kg at container i (calculated using the information given by the sensor in m³ and the material density in kg/m³).
- a_i: Expected daily accumulation rate of container i in kg.
- F_i: Predicted fill level for container i in kg
- E_i: Capacity of container i in kg.
- ψ: Maximum container overflowing threshold in %.

Decision Variables

- x_{ij}: Binary variable indicating if the edge (i, j) is visited, $(i, j \in I)$.
- m_i: Binary variable indicating if waste container i is visited, $(i, j \in I \backslash \{0, n\})$.

Objective Function

$$\min P = C \sum_{i \epsilon I} \sum_{j \epsilon I, (j \neq i)} x_{ij} d_{ij} \tag{1}$$

The objective function aims to minimize the total costs, defined as the total distance travelled multiplied with a cost constant for each kilometer considering costs for fuel, depreciation and labor.

Constraints

Predicted Fill Levels

$$F_i = S_i + a_i, \qquad \forall i \in I \tag{2}$$

The predicted fill levels are calculated based on the current fill level communicated by the sensor and an average arrival rate for that day to address uncertainty of waste accumulation.

$$m_i = \begin{cases} 1 \; if \; F_i \geq \psi * E_i, \\ 0 \text{ otherwise,} \end{cases} \qquad \forall i \in I \tag{3}$$

This constraint ensures that only containers are scheduled for collection that exceed the defined threshold ψ.

Must Visit Constraints

$$\sum_{j \epsilon I, (j \neq i)} x_{ij} = m_i, \qquad \forall i, j \, \epsilon \, I \tag{4}$$

$$\sum_{j \epsilon I, (j \neq i)} x_{ji} = m_i, \qquad \forall i, j \, \epsilon \, I \tag{5}$$

Constraint (4) and (5) ensure that each container which is selected for collection is visited exactly once, resulting in an ingoing and outgoing arc.

Depot leave and return constraints

$$\sum_{i=1}^{n} x_{i0} = 1 \tag{6}$$

This constraint ensures that the vehicle finishes the tour at the depot and returns after the last container.

$$\sum_{j=1}^{n} x_{0j} = 1 \tag{7}$$

This constraint guarantees that the vehicle leaves the depot and sets the depot to the starting point of the tour.

Subtour Elimination

$$u_i - u_j + Q * x_{ij} \leq Q - F_j, \qquad \forall i, j \, \epsilon \, \{1, \dots, n\} \backslash \{i \neq j\} \tag{8}$$

$$0 \leq u_i \leq Q, \qquad \forall i = 1, \dots, n \tag{9}$$

Constraints (8) and (9) represents the Miller-Tucker-Zemlin subtour elimination constraint that prohibits cycles in collection routes.

Vehicle Capacity Constraint

$$\sum_{i=1}^{n} F_i * m_i \leq Q \tag{10}$$

This constraint ensures that the expected amount of waste collected during the collection route does not exceed the capacity of the vehicle.

Variable Domain Constraints

$$x_{ij}, m_i \in \{0,1\}, \qquad \forall i, j \in I, i \neq j \tag{11}$$

This constraint ensures that binary values only are assigned to x_{ij} and m_i.

3.3. Heuristic Design

The heuristic design provides a detailed description of the implementation to solve the stochastic and dynamic inverse IRP for larger instances focusing on practicability since exact solutions become computationally infeasible with increasing problem size. Therefore, a two-stage stochastic optimization model was implemented based on the work of Jorge et al. (2022) aiming to obtain near-optimal solutions within a reasonable amount of time to serve as a daily decision support system. The integrated predictive model addresses stochasticity and enables dynamic routing since only containers are scheduled for collection which are expected to overflow soon aiming to increase cost effectiveness while avoiding overflowing containers and simultaneously increasing service level quality.

3.3.1. Structure of LNS-based Multi-Day Routing Heuristic

The heuristic approach is designed as sequential approach and consists of two phases to solve the medical waste collection problem over a multi-day planning horizon.

In the first phase, a predictive model determines forecasts based on historical data for a defined planning horizon to determine collection schedules for each day of the planning horizon. Containers that exceed a defined threshold are identified as must-go containers and are scheduled for collection on that day they exceed the threshold. This procedure ensures that only containers are visited that need to be emptied aiming to avoid overflowing containers while decreasing total travel distance resulting in increased operational effectiveness. The output of phase one is a detailed schedule for the planning horizon showing which containers must be visited by what date at the latest.

In the second phase, the schedule from phase one serves as a basis for route optimization procedures. Therefore, a Nearest Neighbor and Large Neighborhood Search algorithm are implemented to decrease total travel distances over the planning horizon. In the first step, initial and feasible routes are determined by applying a Nearest Neighbor algorithm for each day while considering capacity constraints. Afterwards, the Large Neighborhood Search algorithm is applied to further optimize initial routes by using destroy and repair operators aiming to shift containers to earlier days and merge routes to decrease total travel distance. It is important to shift containers to earlier days only since they might otherwise overflow. Additionally, also capacity constraints of vehicles are considered to ensure that feasible solutions are obtained by the Large Neighborhood search algorithm. The output phase two is a detailed collection plan

for each day of the planning horizon, whereas the route for the actual day is performed. The structure of the proposed approach can be found in Figure 3-3.

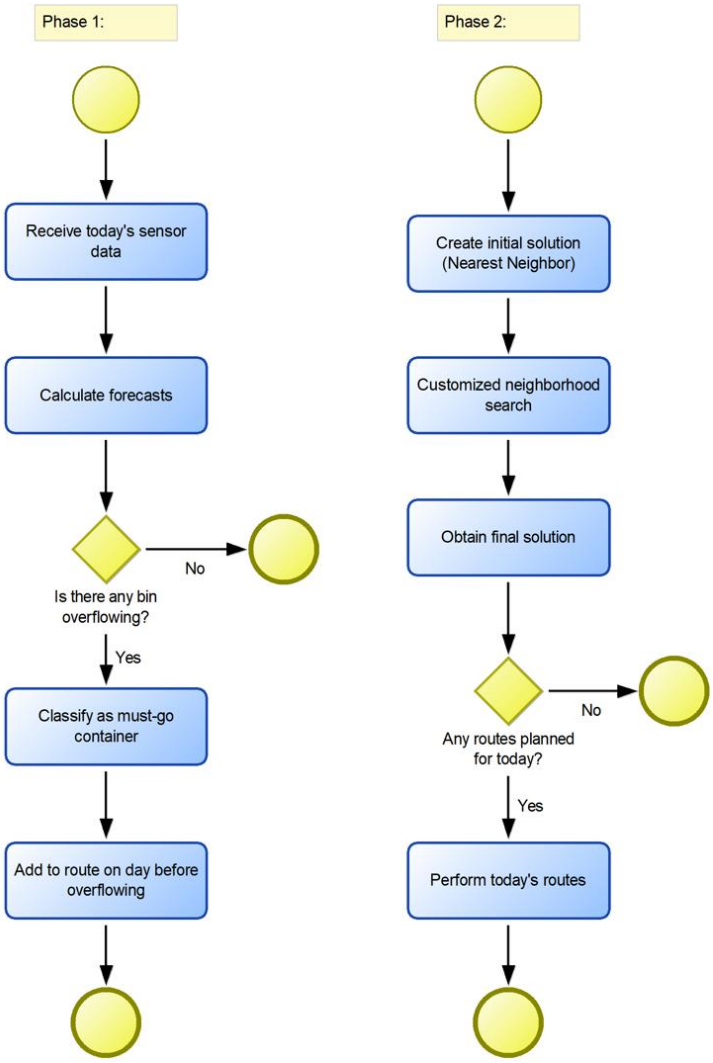

Figure 3-3 Structure of the proposed algorithm

The implementation of the described sequential procedure is outlined by Algorithm 1.

Algorithm 1 Medical Waste Collection

```
Initialize matrix with sensor data from txt-file using Filehandler
Initialize list of containers with corresponding fill levels from sensor data
for every day d do
    Determine forecasts using Forecasting Algorithm (Algorithm 2)
    for every container c do
            Add waste to container c that was sent by sensors this morning
            Update arrival rate of container c accordingly
    end for
    Determine must-go containers by using Must-Go Algorithm (Algorithm 3)
    Create initial solutions using Nearest Neighbour Algorithm (Algorithm 4)
    Run Neighbourhood Search to optimize initial solution (Algorithm 6)
end for
```

3.3.2. Required Adjustments

To solve the dynamic waste collection problem, some customizations were made to the metaheuristic aiming to obtain feasible solutions due to numerous characteristics of the given problem:

- Only containers that are identified as must-go containers can be scheduled for collection. This ensures that only containers are visited that are expected to overflow to decrease total travel distance while maintaining service level quality. Furthermore, this also reduces complexity resulting in decreased runtime.
- The usage of destroy and repair operators is limited, since random operations might result in infeasible solutions, for example if containers are moved to later days than initially scheduled will potentially lead to overflows. Therefore, random operations are prohibited since this would result in increased runtime due to numerous infeasible solutions.
- Since moving containers back in time might lead to overflows, the algorithm must shift containers to earlier days only aiming to avoid infeasible solutions and decreasing runtime.
- Collection routes with low number of containers or waste are not economically viable. Therefore, the algorithm focuses on routes with low number of containers to merge them with other routes aiming to increase operational effectiveness.

3.3.3. Forecasting

The heuristic approach utilizes a predictive model to anticipate future waste fill levels aiming to improve route planning while avoiding container overflows resulting in increased operational effectiveness. Therefore, the predictive model estimates future fill levels based on a combination of historical and real-time data which is transmitted by sensors attached to containers. In this scenario, real-time data ensures that only containers are visited that are expected to overflow resulting in decreased unnecessary mileage and increased cost-effectiveness (Spinelli et al., 2024). Furthermore, if hospitals and pharmacies are only visited when needed, will lead to less distraction of employees so that they can focus on their operational business leading to increased service level quality. Several studies have already demonstrated the effectiveness of using real-time data to significantly increase cost-

effectiveness and improve overall planning performance in the context of waste collection (Cubillos et al., 2022).

Consequently, the keys for efficient waste collection are current and anticipated fill levels to improve proper planning and maximize operational effectiveness. Since waste fill levels are hard to predict, because they are influenced by numerous factors like seasonal effects and customer demand, the waste accumulation rate can be classified as uncertain. Even with real-time sensors the waste accumulation rate for the upcoming days remains uncertain. To address the challenge of uncertainty of waste accumulation, a predictive model should be integrated to anticipate future fill levels based on historical data while considering stochasticity. This ensures that decision-making is based on more informed data to increase quality of route planning.

The implemented predictive model is based on the work of Jorge et al. (2022) which assumes that waste generation follows a stochastic process, approximated by a normal distribution. The model continuously calculates average arrival rates ($\bar{a}_i$) based on historical and real-time data as well as corresponding standard deviation (σ_{ai}) to address uncertainty. The formula for anticipated fill levels can be determined by following formula:

$$\widehat{W}_{it'} = n_{tt'} * \bar{a}_i + \alpha * \sqrt{n_{tt'}} * \sigma_{ai}, \forall i \epsilon B, t' \epsilon [t+1, T]$$

where

$\widehat{W}_{it'}$: estimated fill level in container i at day t';

$n_{tt'}$: number of days between collection day t and day t';

$\bar{a}_i$: average daily waste arrival rate of container i;

α: value for a probability retrieved from the normal distribution; and

σ_{ai}: standard deviation of the daily waste arrival rates of container i.

The average waste arrival rate ($\bar{a}_i$) is maintained in the waste container class as a public integer variable and is updated when waste is added by calling the AddWaste(int newWaste)-method by calculating the mean:

$$\bar{a}_i = \frac{\sum_{j=1}^{n} a_{ij}}{n}$$

where

$\bar{a}_i$: average daily waste arrival rate of container i;

a_{ij}: waste arrival rate of container i on day j; and

n: total number of days.

Algorithm 2 Forecasting Algorithm

1: Read **sensor data** with current fill levels
2: Initialize empty **forecasting-matrix** (number of containers x days of planning period T)
3: **for** each container i **do**
4: **for** each day t' of the planning horizon **do**
5: Calculate the **average arrival rate** $\bar{a}_i$ based on historical data
6: Determine **forecasts** $\widehat{W}_{it'}$ based on average arrival rate and normal distribution
7: Add **forecasts in matrix** for container i and day t'
8: **end for**
9: **end for**
10: **return** forecasting-matrix

Algorithm Explanation

The implemented forecasting algorithm receives sensor data with current fill levels and utilizes this data to determine a forecasting matrix with the size of the number of containers times the days of the planning horizon. Therefore, each row (i) contains the forecasts $(\widehat{W}_{it'})$ for each day t' of the planning horizon for container i calculated by using the formula described above. The forecasting matrix is then handed over to the next step, to identify must-visit collection plans for the planning horizon.

3.3.4. Identification of Must-Go Containers

The last procedure in phase one of the sequential approach is the identification of must-go containers that must be scheduled for collection to avoid overflowing containers aiming to improve service level quality. The implementation of the identification of must-go containers is demonstrated by pseudo code in Algorithm 3.

Algorithm 3 Determine Must-Go Containers

1: Initialize empty matrix for **must-go schedule** (days of planning horizon X number of containers)
2: Define **critical threshold** for must-go container
3: Read forecasting-matrix
4: **for** each container of forecasting matrix **do**
5: **for** each day of planning horizon **do**
6: **if** expected fill level is equal or greater than critical threshold **then**
7: **if** container has not been scheduled already **then**
8: **Schedule** visit for container on corresponding day
9: **end if**
10: **end if**
11: **end for**
12: **end for**
13: **return** must-go matrix

Algorithm Explanation

Based on the forecasting matrix from the predictive model, the algorithm determines must-go containers that need to be visited by collection vehicles. To do so, the code iterates for each container through all days of the planning period and checks whether the anticipated fill level exceeds the defined critical threshold and if so, it will be scheduled for collection. This pre-processing ensures that the number of containers that must be considered during route optimization is decreased to reduce complexity and runtime. The identification of must-go containers builds a bridge between forecasting and route optimization and is therefore crucial for operational effectiveness, since visiting containers too early will lead to more visits and unnecessary mileage whereas visiting too late might result in overflowing containers if forecasts are not precise enough. Consequently, the key for efficient dynamic routing is to find a balanced critical threshold to optimize medical waste collection.

3.3.5. Construction Phase

After determining must-go containers based on forecasts from historical and real-time data, initial routes must be created which serve as a basis for subsequent route optimization procedures. For initial routing, a greedy heuristic is implemented to determine feasible routes while maintaining good solution quality and computational efficiency. Therefore, a Nearest Neighbor algorithm is implemented which is well suited for quickly generating feasible solutions for vehicle routing problems. Alsalibi et al. (2012) demonstrated that Nearest Neighbor algorithm provides near-optimal and also optimal solutions for small datasets. Since especially initial routes tend to be smaller as they have not been merged by the Large Neighborhood Search algorithm, the usage of Nearest Neighborhood algorithm can lead to promising results for initial routing and therefore provides an efficient base for subsequent route optimization. The implementation of this procedure is demonstrated by Algorithm 4.

Algorithm 4 Nearest Neighbour

```
1:  Read initial tour of the day
2:  Initialize list with unvisited containers
3:  Initialize nearest neighbour tour with depot as starting point
4:  while not all containers are visited do
5:      Initialize cheapest tour with Max Value
6:      for each container in unvisited containers do
7:          Calculate costs from last container of nearest neighbour route to unvisited
            container
8:          if calculated costs are smaller than cheapest tour then
9:              set container as cheapest option
10:             set cheapest tour to current tour costs
11:         end if
12:     end for
13:     Add cheapest container to nearest neighbour route
14:     Remove cheapest container from unvisited container list
15: end while
16: return nearest neighbour route
```

Algorithm Explanation

The implementation of the Nearest Neighbor algorithm first receives the must-go matrix and creates an initial tour for each day accordingly. For each route it initializes a list with all containers that must be visited on that day, representing a list of unvisited containers. Then it initializes a second list starting with the depot, representing the route. Then it iteratively determines the nearest unvisited container to the last container of the route. After the nearest container has been identified, it will be added to the route and is deleted from the list of all unvisited containers. This procedure is reperformed until the list of all unvisited containers is empty. At the end, the depot is appended to the end to guarantee that each route starts and ends at the depot. The whole process is then reperformed for each day of the planning horizon.

The routes obtained then serve as a basis for further route optimization by the Large Neighborhood Search algorithm aiming to minimize total travel distance while avoiding container overflows to increase service level quality.

3.3.6. Large Neighborhood Search

Since initial routes have already been created, a route optimization algorithm should be applied to further increase cost-effectiveness. Therefore, a Large Neighborhood Search algorithm (LNS) is implemented which is a powerful metaheuristic that destroys current solutions and rebuilds them by applying selective reinsertions to explore new solution spaces and escape local optima.

Neighborhood Search

In general, neighborhood search algorithm is an optimization method that iteratively improves a solution by exploring its neighborhood. In every iteration it determines the best solution and moves on until no improvement can be found and a local optima is reached. This approach is also known as the steepest descent algorithm (Pisinger & Røpke, 2010).

Large Neighborhood Search

The LNS was initially proposed by Shaw (1998), consisting of two major steps. First an initial solution gets partially destructed by a destroy operator, followed by a repair operator that rebuilds it and creates a new solution. The destroy operator usually involves randomness to ensure that different parts of a solution are destructed in each iteration to explore new neighborhoods.

To illustrate how the LNS algorithm works, a destroy operator could randomly remove 15% of the customers of a CVRP whereas the repair operator iteratively inserts each removed customer at the cheapest position. This procedure can be seen in Figure 3-4. The initial solution is shown in the top left figure, followed by the destructed solution after removing 15% of the customers in the top right figure. The new neighborhood solution after applying the repair operator by using cheapest insertion is shown in the bottom figure (Pisinger & Røpke, 2010).

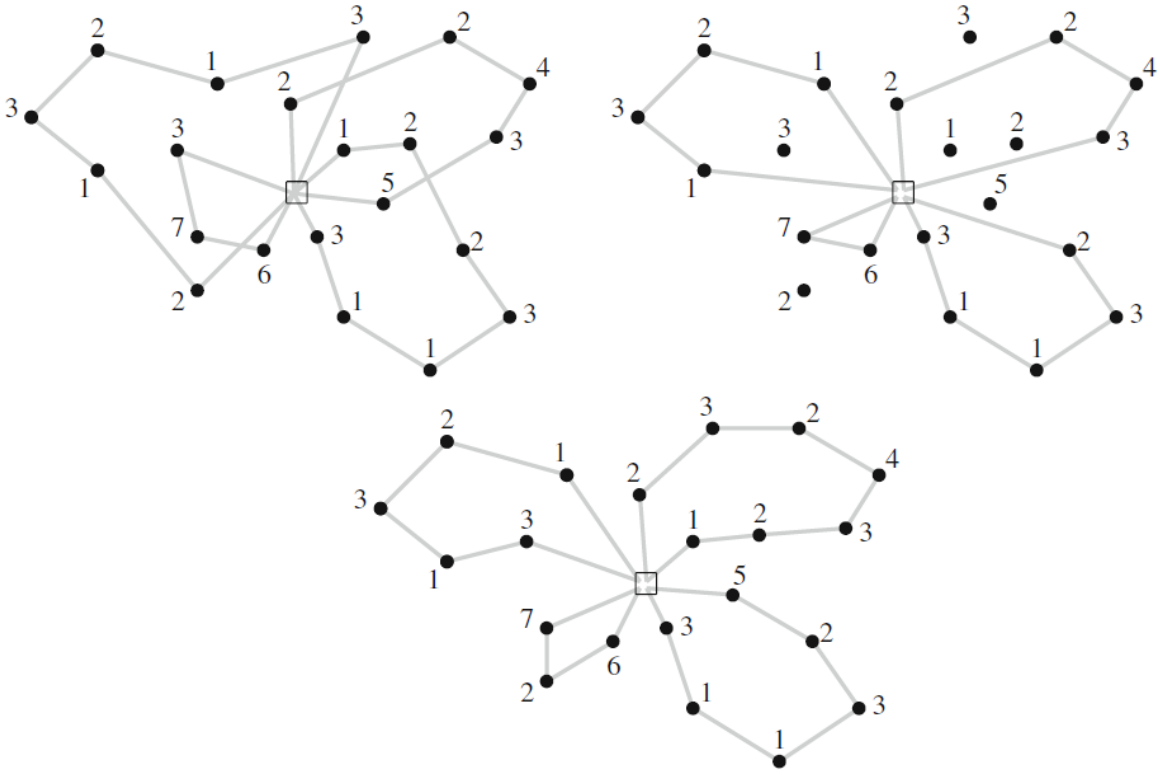

Figure 3-4 Example of destroy-and-repair method (Pisinger & Røpke, 2010)

According to Pisinger & Røpke (2010), the implementation of a LNS algorithm in general can be performed as shown in Algorithm 5.

Algorithm 5 Overview of Large Neighborhood Search

```
input: a feasible solution x
x^b = x;
repeat
    x^t = r(d(x));
    if accept (x^t, x) then
        x = x^t;
    end if
    if c(x^t) < c(x^b) then
        x^b = x^t;
    end if
until stop criterion is met
return x^b
```

Algorithm Explanation

In general, the LNS algorithm maintains three solutions. First, the best solution found so far represented by x^b, a current solution x and a temporary solution x^t, which will be determined as the best solution so far or discarded otherwise. Furthermore, the destroy operator is

represented by $d(x)$ and the repair operator is represented by $r(d(x))$, meaning that the destroyed initial solution gets reconstructed.

The algorithm starts with initializing the best solution with the current solution in line 2. Afterwards the iterative destroy and repair procedure starts to obtain a new temporary solution x^t in line 4. Afterwards the algorithm determines whether the solution can be accepted as new current solution x, for example if it leads to an improved solution. Then line 8 checks if it is also better than the best solution known so far. If so, the temporary solution x^t will be set as the new best solution x^b. This procedure is reperformed after a defined stopping criteria is reached. A stopping criteria can be a fixed number of iterations, elapsed time or other individually defined criteria.

Implementation of LNS

To solve the stochastic and dynamic waste collection problem, several constraints must be considered such as containers must only be moved to earlier days of the planning horizon to avoid container overflows, the fact that random operations potentially increase runtime and capacity constraints of vehicles. Those complex constraints drastically limit the number of feasible search moves, as many operations will result in infeasible solutions. Consequently, this leads to a limited search space making traditional local search move operations less effective. To tackle this challenge, a LNS algorithm is implemented since Shaw (1998) demonstrated that LNS is well suited to tackle this challenge, since it provides far-reaching move operators that allows to move over barriers created by constraints and creates new feasible solutions.

Based on that information, the implementation of a LNS algorithm presents a promising approach to tackle route optimization for the stochastic and dynamic waste collection problem aiming to minimize total travel distance while avoiding container overflows to increase service level quality. The pseudo-code for the implementation of the LNS algorithm is shown in Algorithm 6.

Algorithm 6 Large Neighbourhood Search Implementation

```
Read nearest neighbour routes
Initialize cheapest tour costs
Initialize previous tour costs based on current nearest neighbour route
for each iteration until stop limit is reached do
    Determine route with least amount of total waste
    for each container c in smallest route do
        for each day d before container c is initially scheduled do
            if enough vehicle capacity is available on a day d then
                Schedule the container c on day d
                break
            else
                Container c cannot be scheduled earlier
            end if
        end for
    Determine cheapest insertion index for container on scheduled day
    Insert container on scheduled day at cheapest insertion index
    Remove container from initial route
    end for
end for
Calculate new tour costs
if current tour costs are smaller than previous tour costs then
    set current tour as cheapest tour
end if
```

Algorithm Explanation

As outlined in this section, the LNS algorithm consists of two major components, the destroy and repair operator. Since the medical waste collection problem requires some adaptions in terms of constraints, some modifications are made to obtain feasible solution within reasonable amount of time. The input for the LNS is provided by the Nearest Neighbor algorithm, where initial routes for each day of the planning horizon are created.

- **Destroy Operator**

  The destroy operator plays a crucial role in the LNS algorithm since the amount of destruction significantly affects the solution quality and efficiency of the algorithm. If the degree of destruction is too little, the heuristic may be faced with issues in exploring the search space and the effect of LNS gets lost. If the degree of destruction is too high, the process will result in repeated re-optimization which can be time consuming and might lead to poor solution-quality (Pisinger & Røpke, 2010).

  The destroy operator is designed to effectively destroy routes such that they can be merged into other routes aiming to decrease total travel distance while avoiding container overflows. Since small routes have the highest possibility to be merged into

other routes while meeting capacity constraints, the destroy operator focuses on small routes. Furthermore, small routes might be economically not viable since the vehicle have to leave and return to the depot and if such route is consolidated, unnecessary mileage can be avoided. Therefore, the destroy operator identifies the route with the lowest waste volume and removes the containers from the route. This approach aims to find a balanced degree of destruction, by destroying potentially inefficient routes but not the whole solution.

The destroy operator was designed to consider the required adjustments from section 3.3.2 which demonstrates that the chosen LNS is well suited to tackle the challenges of dynamic medical waste collection:

- The Nearest Neighbor Matrix only contains must-go containers which reduces complexity and runtime of the algorithm aiming to provide feasible solutions within reasonable amount of time also for large problem sizes.
- The destroy operator focuses to eliminate economically inefficient routes to increase overall cost-effectiveness.

- **Repair Operator**

The implementation of a repair function in LNS offers flexibility, since it can be chosen between optimal and heuristic solutions. However, it should be noted that an optimal solutions will take much longer than a heuristic solution. Additionally, optimal solutions will only produce improving or identical solutions, making it difficult to leave local optima unless a large part of the solution gets destroyed. Therefore, the implementation of an optimal solution is not attractive (Pisinger & Røpke, 2010).

The repair operator is designed to effectively insert the containers of the smallest route determined by the destroy operator, such that they can be consolidated to decrease total costs and while considering capacity constraints. To ensure that a removed container will not overflow, they can be inserted into routes on earlier days only. To evaluate the cheapest position for insertion, the algorithm explores all routes of earlier days in the planning horizon and determines the cheapest insertion index. This approach focuses on ensuring result-oriented reinsertion rather than random reconstruction aiming to reduce runtime while providing good solution quality.

Also during the implementation of the repair operator required adjustments as described in section 3.3.2 were considered:

- Random re-insertion operations are avoided, since this potentially increases the number of infeasible solutions and consequently results in increased runtime and container overflows. Also Shaw (1998) highlighted that selected re-insertions significantly led to better results than random selection.
- The removed containers must be inserted into routes on earlier days of the planning horizon.

- **Acceptance Criterion**

  In each iteration the new solution obtained from the repair operator is validated by an acceptance function and only gets accepted if the new solution results in decreased total costs.

- **Stopping Criterion**

  The algorithm stops after a defined number of iterations which is dynamically determined based on the given problem size.

4. Data Analysis and Results Discussion

4.1. Introduction

This chapter presents the experimental evaluation of the proposed two-phase hybrid optimization approach, as well as the analysis of a traditional static approach. The simulation is applied on real-world data to ensure representative results that can be used to further discuss practical relevance of the proposed approach. Consequently, the results from the dynamic and static routing strategy are then compared in terms of cost-efficiency and robustness, aiming to answer the central research question: *How to enhance cost-effectiveness of medical waste collection by using real-time data provided by sensors?*

The chapter first outlines the simulation objectives and the assumptions made for the simulation environment so that general conditions are defined to establish a sound framework for the evaluation of the static and dynamic routing strategy. Afterwards, the computational setup is presented followed by the dataset generation which describes the source of real-world data and how it is used to evaluate the performance of the proposed dynamic approach compared to the traditional approach. Then, the accuracy of the predictive model for the dataset applied should be analyzed, as it plays a vital role for the bin selection of future periods and therefore affects efficient route planning. To obtain the best configuration of the dynamic approach, a parameter tuning is performed to further optimize cost-effectiveness by adapting the planning horizon and container threshold level. Afterwards, the evaluation of the static and dynamic approach is performed. To obtain exact solutions, a simulation-optimization approach is then performed that feeds the forecasts from the predictive model into the mathematical model aiming to solve the given problem. In the next step, the results are compared to those from the static routing approach to answer the main research question.

4.2. Simulation Objectives and Assumptions

This section provides an overview of the simulation objectives that are defined to answer the research questions from chapter 1.2. The main objective is to determine whether the proposed dynamic approach increases cost-effectiveness compared to a traditional static routing strategy resulting in an efficient tool for decision-making. To ensure that both strategies are tested under identical conditions, assumptions are made to establish a framework for simulation in which both approaches are being tested.

4.2.1. Simulation Objectives

To answer the main research question: *How to enhance cost-effectiveness of medical waste collection by using real-time data provided by sensors?,* the assessment of the proposed two-phase approach is decomposed into its two phases. In the first step, phase one is evaluated by studying the accuracy of the predictive model as it affects decision-making to timely collect containers. Then the overall performance of the dynamic approach with different amounts of containers will be evaluated to compare the results to a traditional static approach. To achieve these results, following simulation objectives are defined:

- To assess the suitability of the integrated predictive model for timely collection decision making by applying real-world data.
    - This objective emphasizes empirical testing to evaluate its accuracy in forecasting container fill levels that result in improved route planning and avoiding unnecessary mileage.
- To quantify the rate of container overflows for both routing strategies to determine robustness of each approach.
    - This objective aims to determine whether the novel approach avoids overflowing containers which decreases health risks to society resulting in a more robust approach and serves as an indicator for service level quality.
- To evaluate the truck utilization during simulation to assess the efficiency of route planning.
    - This objective aims to quantify and compare the efficiency of performed collection routes.
- To quantify the performance of a traditional static routing approach and the proposed dynamic approach in terms of cost-effectiveness by applying identical datasets and simulation framework.
    - This objective contributes to answering the main research question related to cost-effectiveness by quantifying the performance in terms of total travel distance.

4.2.2. Simulation Assumptions

To ensure a controlled analysis of both routing strategies, the simulation is based on a set of simplifications in terms of assumptions. The assumptions outline the scope and the constraints

applied to the static routing strategy as well as to the dynamic approach and ensure reproducibility and comparability.

Assumptions related to operations

The collection network consists of a fixed number of medical waste containers with limited capacity.

Each tour starts and ends at a central depot. Before entering the depot, a vehicle disposes the waste, so that full capacity is available for the next tour.

Vehicle and Routing Constraints

There is a homogeneous fleet, meaning that all vehicles are of the same type with identical characteristics like load capacity and overflows are not allowed.

Travel distances are asymmetrical and pre-calculated in a distance matrix.

Assumptions related to Forecasting

The predictive model determines forecasts based on historical data in combination with a stochastic process modeled by a normal distribution.

Containers exceeding a predefined fill level are classified as must-go containers.

Static and Dynamic Routing Assumptions

The static routing approach follows a fixed schedule, independent of actual or forecasted fill levels.

The dynamic routing approach recalculates collection schedules daily, using the predictive model to identify must-go containers. Based on that, route optimization heuristics are applied to determine optimal or near-optimal collection routes.

These assumptions establish a structured simulation environment for testing both strategies ensuring that performance differences are fully attributable to the routing strategies and not to uncontrolled or random external factors.

4.3. Computational Setup and Dataset Generation

This section provides an overview of the computational setup that is used for running the LNS-based Multi-Day Routing Heuristic for Predictive Medical Waste Collection, followed by a detailed description of the dataset used for simulation and comparison of the static and dynamic routing strategy.

4.3.1. Computational Setup

The proposed LNS-based Multi-Day Routing Heuristic is implemented in C# using standard libraries. The code was developed using Microsoft Visual Studio as integrated development environment, since it provides a wide set of tools for code editing, analysis and debugging. The implementation and all computations were performed on a Lenovo ThinkPad T490s equipped with an Intel(R) Core(TM) i5-8265U 64-Bit machine operating at a clock speed of 1.6GHz, 16.0GB of RAM and running Windows 11 Pro as operating system.

4.3.2. Dataset Generation

The data considered in this study are based on real-world data published by Spinelli et al. (2024) and provided by a Portuguese waste management company. The dataset includes the sensor data of 121 containers for plastic and metal waste and is split into six intervals: 0%, 12.5%, 37.5%, 62.5%, 87.5%, and Overflow. For example, the information of 12.5% means that the current fill level is between 0.1% and 12.5%. Data entries are only available for days on which a container was physically visited, with the collection period spanning April to July 2024. The second source of real-world data was published by de Morais et al. (2024) providing volumetric sensor data for undifferentiated waste from October 2017 until the end of July 2018 for 18 containers which can be used for small-instance tests and especially for testing the accuracy of the predictive model, as it offers daily measurements. Since undifferentiated waste is characterized by a high daily filling rate resulting in high arrival rates (de Morais et al., 2024), this data is used to simulate the aspect of robustness by reconstructing epidemics with extraordinarily high arrival rates. The dataset from Spinelli et al. (2024) is used to examine the model's behavior under regular conditions.

For simulation experiments, two categories of instances are considered, experiments with small instances and those with large instances.

- **Small-instance experiments:** A set of 18 instances is provided by dataset from de Morais et al. (2024). Small-instance experiments allow controlled simulation and faster computational results, enabling parameter tuning.
- **Large-instance experiments:** A set of 50 randomly chosen bins from the dataset provided by Spinelli et al. (2024) will be considered for large-instance experiments to study the behavior of the proposed dynamic approach compared to small-instance experiments to draw conclusions in terms of scalability and practical applicability. Spinelli et al. (2024) also used 50 containers for simulation, since fifty is the average collection size of one route from the industrial partner.

This combination of small- and large-instance experiments with real-world data enables simulation of a realistic environment to study the cost-effectiveness and robustness for several problem sizes and consequently drawing conclusions about practical applicability. Additionally, together with the assumptions from section 4.2.2 the simulation is also reproducible which ensures that differences in results are clearly attributable.

4.4. Analysis of forecasting method

This section presents the performance analysis of the integrated predictive model which is used in the first phase of the two-stage stochastic optimization model to determine must-go containers. The performance analysis aims to quantify the accuracy of the forecasts produced by the model, as they are considered during decision-making for selecting overflowing containers which are then scheduled for collection. Consequently, the predictive model affects the cost-effectiveness of the dynamic approach, since incorrect container selections might increase the number of scheduled containers and therefore also increase the total travel distance by simultaneously decreasing the service level quality.

4.4.1. Analysis of Arrival Rates

The analysis of the accuracy is performed on three randomly selected containers from the dataset published by de Morais et al. (2024), since it provides daily sensor data from fill levels. To obtain the daily arrival rates, which play a vital role for assessing the accuracy of the predictive model, the difference between the fill level of each day was calculated. Based on the calculated arrival rates, the underlying distribution can be studied in the first step to determine

whether the assumption regarding the normal distribution for the implementation of the predictive algorithm is valid.

Visualization of Daily Arrival Rates

The arrival rates for each container and day are visualized in a diagram, with the X-axis representing the days and the Y-axis representing the amount of waste added in percentage of total capacity. The distribution is determined by visualizing the arrival rates in a histogram with ten intervals in Excel with the X-axis representing the intervals and the Y-axis depicting the number of occurrences in the corresponding interval. This is done for the three randomly selected containers with ID 603, 606, and 2309.

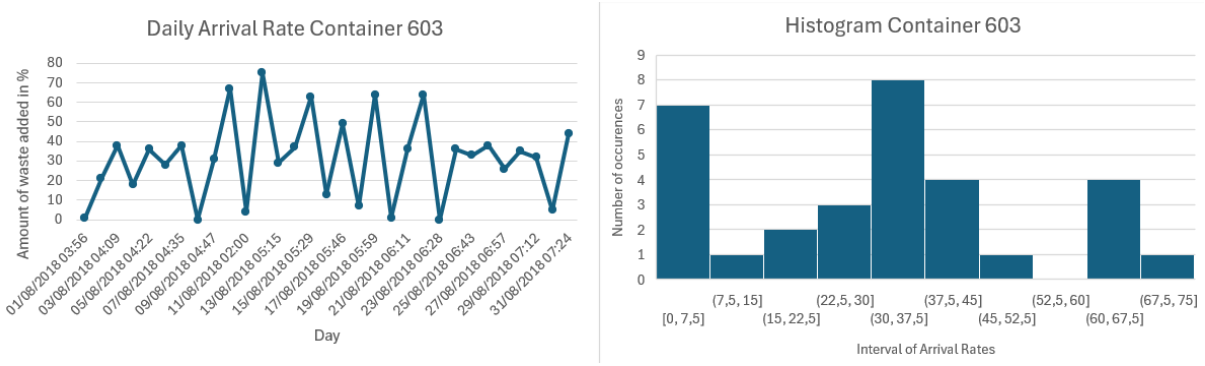

Figure 4-1: Visualization of the daily arrival rates of container 603 derived from the sensor data

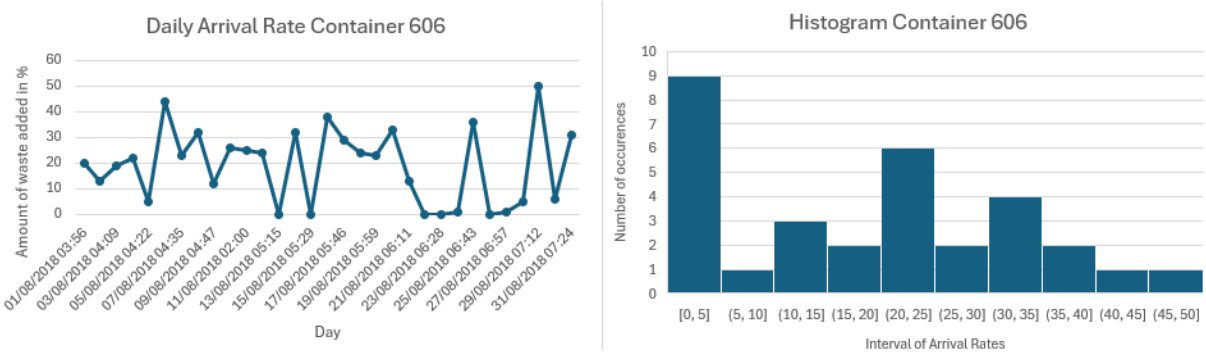

Figure 4-2: Visualization of the daily arrival rates of container 606 derived from the sensor data

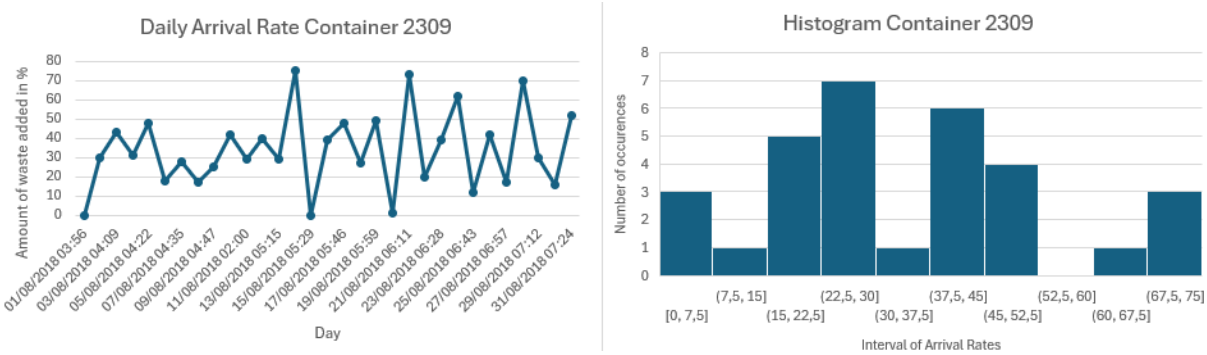

Figure 4-3: Visualization of the daily arrival rates of container 2309 derived from the sensor data

Discussion of Daily Arrival Rate Visualizations

The histograms in Figure 4-1, Figure 4-2, and Figure 4-3 clearly show a bell-shaped curve indicating that the daily arrival rate follows a normal distribution besides some outliers which can be caused by errors in measurement from the sensor, for example if waste is thrown in the container and piles up because it does not distribute evenly since the waste is not liquid. This marks a fundamental finding, as the integrated predictive model also anticipates future container fill levels based on a normal distribution to address the stochastic nature of arrival rates.

The obtained result from the analysis of arrival rates lays a foundation for the design of a precise forecasting model since the underlying distributions are aligned.

4.4.2. Assess the accuracy of the predictive model

After analyzing the input data, experiments are performed to assess the accuracy of the predictive model by applying a dataset of three randomly selected containers with ID 603, 606, and 2309 to the implementation which anticipates the arrival rates based on historical data that is depicted by the average arrival rate and standard deviation to address the uncertainty. Since the analysis of arrival rates indicated that they follow a normal distribution, as well as the implemented predictive model, this finding serves as a promising starting point for assessing the accuracy.

Additionally, the analysis aims to determine whether the accuracy of forecasts improves with increasing amount of historical data. To analyze the level of learning, the forecasts after 5, 10, and 15 days are compared with the actual arrival rates. Based on the result, another fundamental aspect, the length of planning horizon is evaluated, for which the evolution of forecasting errors

is examined to find the optimal configuration for the predictive model. During simulation the length of the planning horizon is set to 15 days.

Visualization of Forecasts compared to actual arrival rates

The results are then visualized in a chart with lines and markers representing the data for each day, where the X-axis represents the days and the Y-axis depicts the corresponding amount of waste arrival for each day. The blue line always represents the actual arrival rate provided by sensor data and the orange line illustrates the forecasts anticipated by the predictive model. To study the aspect of learning, experiments are performed at different time periods, specifically in periods 0, 10, and 15 with a planning horizon of 15 periods.

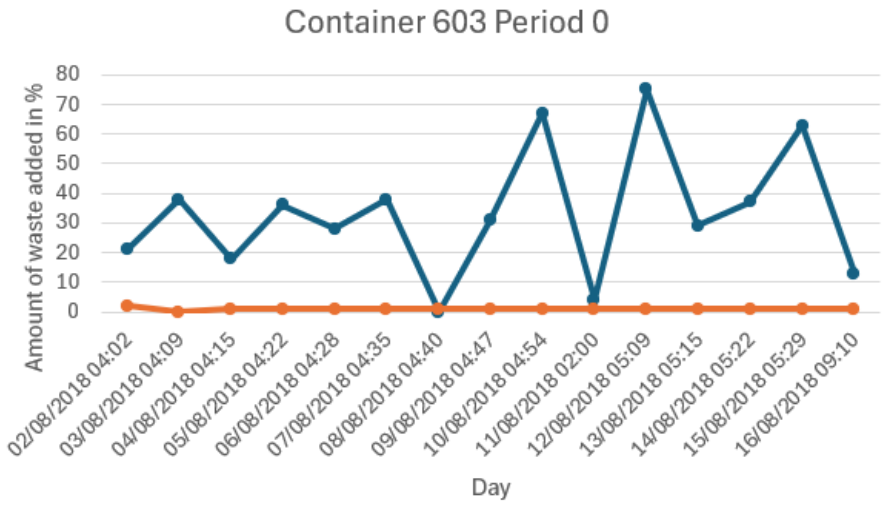

Figure 4-4: Comparison for container 603 in period 0

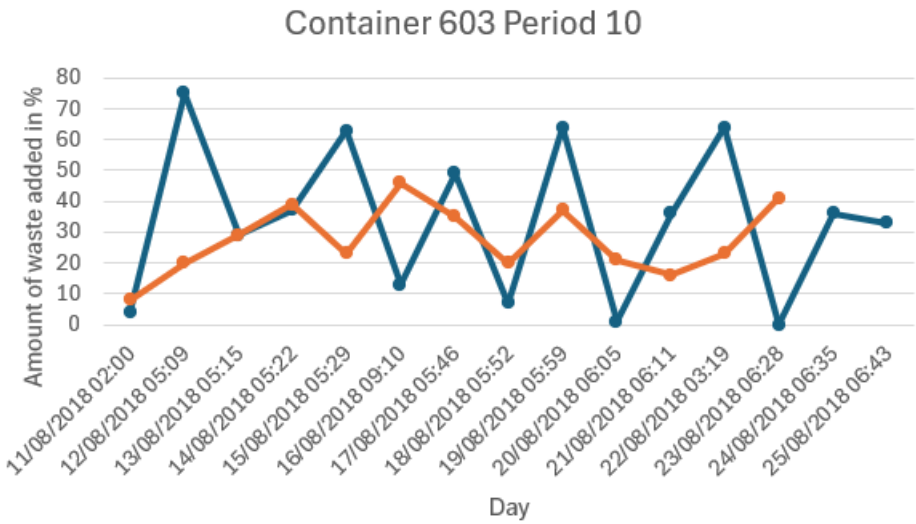

Figure 4-5: Comparison for container 603 in period 10

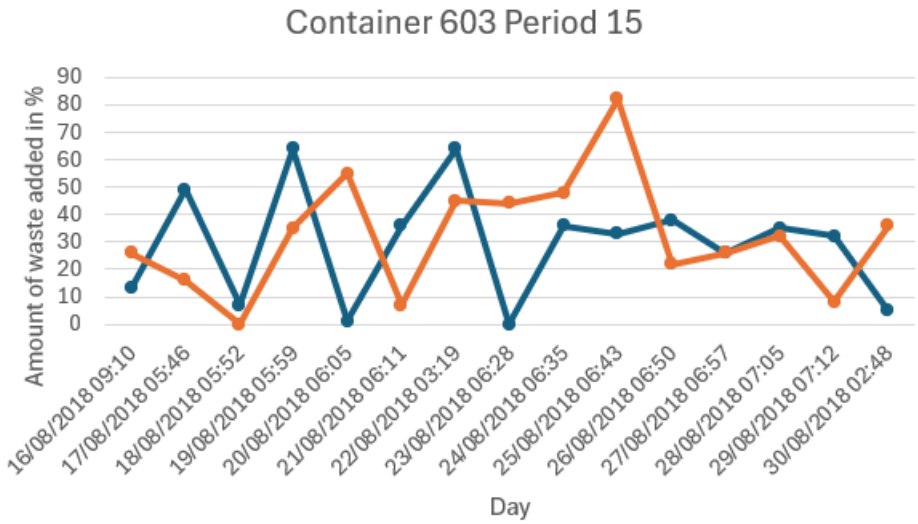

Figure 4-6: Comparison for container 603 in period 15

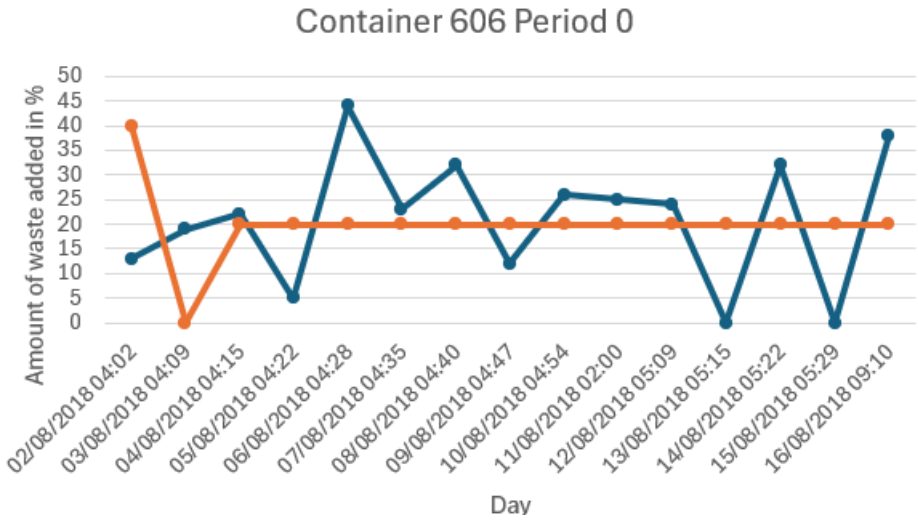

Figure 4-7: Comparison for container 606 in period 0

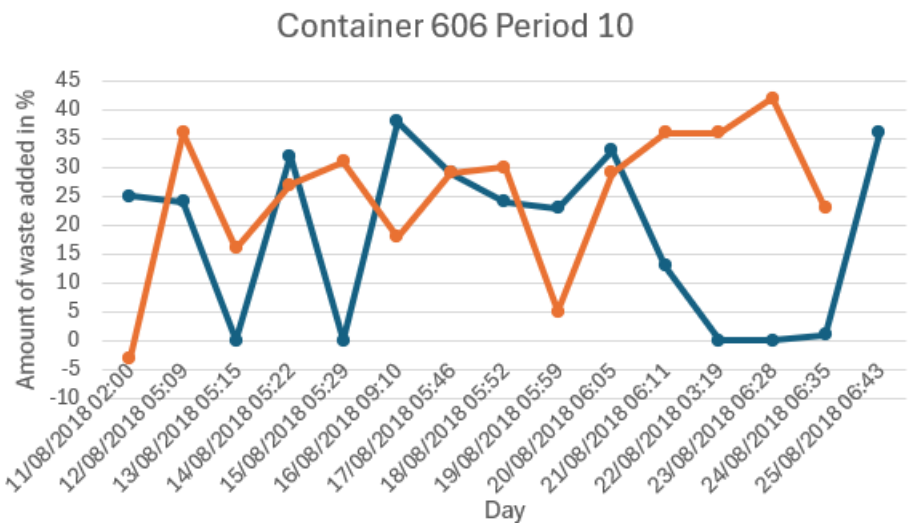

Figure 4-8: Comparison for container 606 in period 10

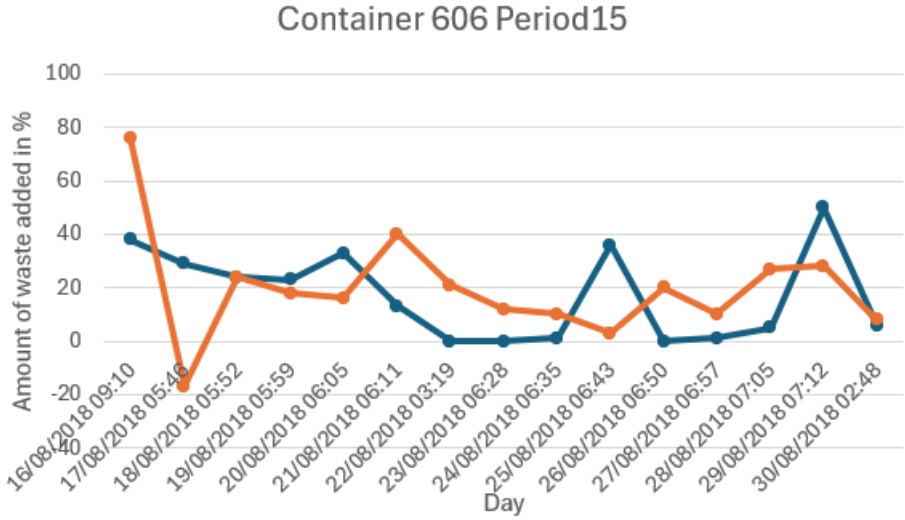

Figure 4-9: Comparison for container 606 in period 15

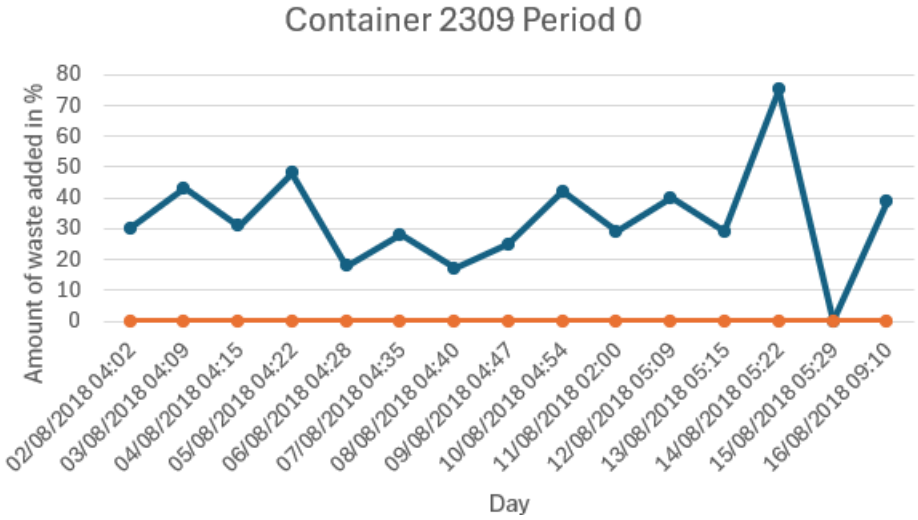

Figure 4-10: Comparison for container 2309 in period 0

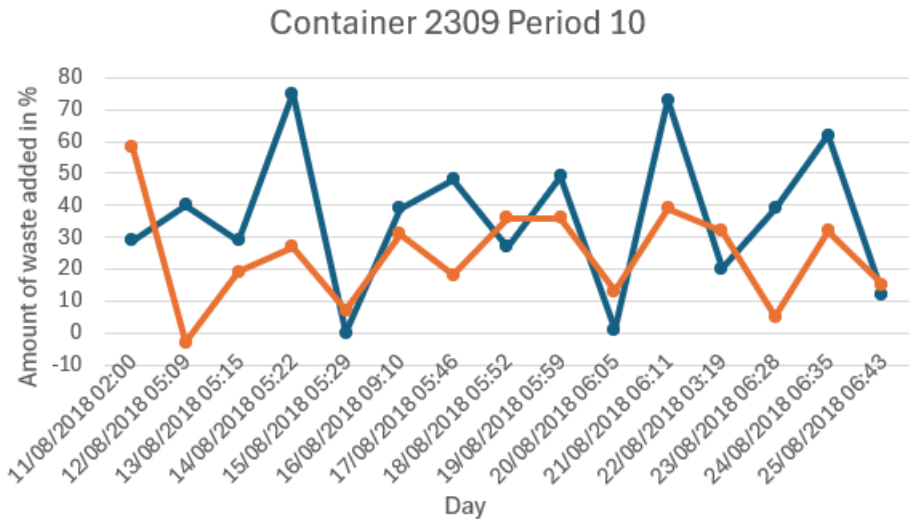

Figure 4-11: Comparison for container 2309 in period 10

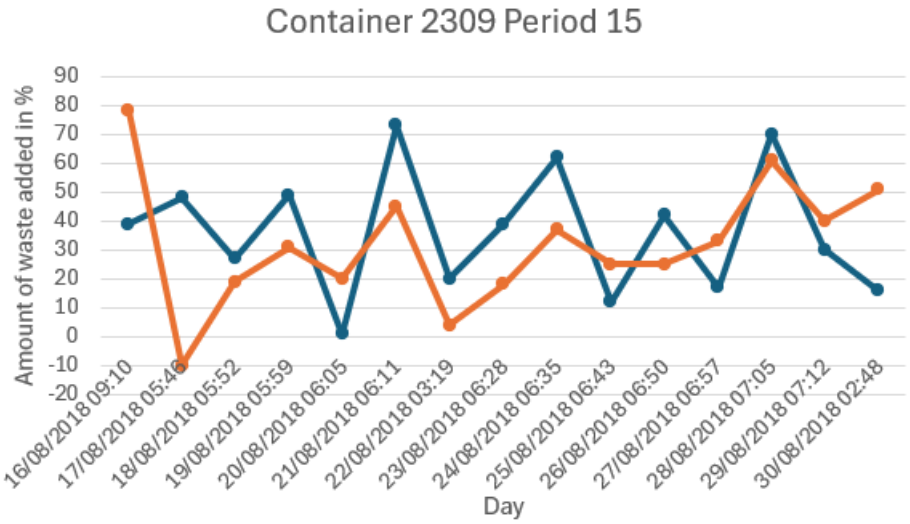

Figure 4-12: Comparison for container 2309 in period 15

Discussion of comparison between sensor data and forecasts

The experiments provide a detailed breakdown of how the accuracy of forecasts evolve over time. Especially in period 0, illustrated by Figure 4-4, Figure 4-7, and Figure 4-10, there is a huge gap between actual sensor data and forecasts indicating very imprecise forecasts since no historical data is available at that point in time. After 10 periods, the forecasts in Figure 4-5, Figure 4-8, and Figure 4-11 are getting more precise, as historical data can be used from the predictive model to anticipate future arrival rates. Finally, Figure 4-6, Figure 4-9, and Figure 4-12 illustrate a comparison after 15 periods, showing almost identical curves with slightly offset in time.

Since the experiments show that the predictive model improves accuracy over time, the model learns with time passing by due to the historical data that becomes available. Based on this finding, subsequent steps can be derived to increase accuracy at the beginning of the simulation, for example using training data before starting the simulation. Furthermore, the results obtained confirm the initial assumption that an aligned distribution of the predictive model and sensor data lead to accurate results.

The experiments not only show that the forecasts become more accurate after 10 to 15 periods but also allow conclusion to be drawn regarding the length of the planning horizon. Based on the data obtained from the experiments, a forecasting horizon of 7 to 9 days represents a well-suited tradeoff between accuracy and length.

The analysis of arrival rates and the corresponding alignment of the predictive model to a normal distribution contributed to obtaining quite accurate forecasts and helped to optimize the configuration of the two-phase hybrid optimization approach. These findings serve as a basis for optimal decision making of route optimization in the second phase to increase overall performance.

4.5. Performance Analysis of Routing Strategies

This section presents a performance analysis of the two-phase hybrid optimization approach and traditional static routing strategy. The results from simulations are then used to perform a comparative performance analysis based on defined performance metrics outlined in this section. To address the stochastic nature of this problem, experiments are performed multiple times to ensure a fair comparison. The performance analysis aims to answer the main research question whether dynamic routing increases cost-effectiveness compared to static strategies as well as determining the practical applicability by studying the behavior of the proposed approach for larger instances to examine scalability and how it affects solution quality.

4.5.1. Evaluation Metrics and Framework

To quantify and systematically rank the performance of each routing strategy, evaluation metrics are defined as follows:

- **Total Travel Distance:** Measures the total distance travelled by a vehicle in meters or kilometers. This metric is essential to quantify the overall performance of routing

strategies, as it is a main cost factor and provides insights on how efficiently the routing algorithm works.

- **Total costs:** To determine cost-effectiveness, the total travel distance will be considered since it is the main cost-driver and affects other costs like fuel consumption, vehicle depreciation and labor costs directly. Therefore, other cost drivers will be neglected.
- **Rate of overflowing containers:** The rate of overflowing containers represents the ratio of containers exceeding their capacity to the number of all containers considered for simulation. This metric is an important aspect for determining service level quality and consequently the practical applicability and robustness of a routing strategy. Furthermore, it allows assumptions to be drawn about the health risk society is faced with since medical waste may transmit diseases when getting in touch with it.
- **Average vehicle utilization:** This metric measures the ratio of how much of the available truck capacity was used per trip on average during the simulation and allows assumptions to be made about the efficiency of a routing strategy since low utilization might be an indicator for unnecessary mileage.

$$\textbf{Vehicle utilization} = \frac{\textbf{amount of waste carried in tour}}{\textbf{total truck capacity}}$$

The defined routing metrics serve as an enabler for quantifying performance of routing strategies and allow a systematic comparison of different approaches in terms of overall cost effectiveness and practical applicability.

4.5.2. Small-Scale Experiments

To assess the behavior of static and dynamic routing strategies, experiments on small-scale instances of 18 containers are performed on real-world data. Both routing strategies are tested by applying the same dataset ensuring that differences are solely attributable to a routing strategy. The data was published by de Morais et al. (2024) and provided by a Portuguese waste collection company. The vehicle capacity is restricted with 2.000kg, containers have a volume of 1m³ and waste density is 123,2 kg/m³.

Since an initial data analysis showed that arrival rates are relatively high compared to the container capacities, a second test is performed with an increased container capacity by one third, leading to a proportional reduction of arrival rates to simulate also cases with lower amounts of waste disposal. This is done to study the behavior of the proposed approach under

several circumstances including periods with high and low waste arrival rates to draw conclusions about its robustness and flexibility.

Static Routing Strategy

Static routing strategies perform container collections regardless of current fill levels on predefined days. To assess the performance of a static routing strategy, the schedule from the Portuguese waste collection company is applied. In September 2017, the waste collection company equipped 18 containers with sensors while maintaining a static routing strategy, where all equipped containers are visited every Monday, Thursday, and Saturday (de Morais et al., 2024). This schedule is also applied for the simulation of a static routing strategy on small-scale instances. As sensor data is available from September 2017 until August 2018, one month – August 2018 – was randomly selected for the simulation.

For the evaluation of the static routing strategy, the corresponding routes were determined based on the provided data to calculate the evaluation metrics total travel distance, rate of overflowing containers, and vehicle utilization.

Table 4-1: simulation results of static routing strategy

Date	Weight (kg)	Distance (m)	Collected bins	Overf. bins (Si≥100%)	Overf. ratio (%)	Vehicle Utilization (%)
Thursday, 2 August 2018	1875,10	316 127,24	18	12	66,67	93,76
Saturday, 4 August 2018	1558,48	311 650,33	18	5	27,78	77,92
Monday, 6 August 2018	1352,74	311 660,25	17	6	33,33	67,64
Thursday, 9 August 2018	1892,35	426 932,02	18	10	55,56	94,62
Saturday, 11 August 2018	1517,82	420 956,44	18	6	33,33	75,89
Monday, 13 August 2018	1479,63	330 680,75	18	8	44,44	73,98
Thursday, 16 August 2018	1538,77	354 370,29	17	11	61,11	76,94
Friday, 17 August 2018	86,24	19 853,36	1	0	0,00	4,31
Saturday, 18 August 2018	1450,06	346 466,04	17	6	33,33	72,50
Monday, 20 August 2018	1514,13	373 436,37	18	6	33,33	75,71
Thursday, 23 August 2018	1807,34	296 115,54	17	12	66,67	90,37
Saturday, 25 August 2018	1196,27	331 771,79	17	4	22,22	59,81
Monday, 27 August 2018	1378,61	417 117,09	18	5	27,78	68,93
Thursday, 30 August 2018	1918,22	360 813,08	18	12	66,67	95,91
Total Sum	20 565,78	4 617 950,59		Average	40,87	73,45

The results show a total travel distance of approximately 4.618km in August 2018. The total travel distance was calculated based on the routes that were performed on these days, as every collection visit has a timestep which allows to derive detailed collection schedules. The results

of the evaluation metrics indicate a low service level quality, since the average rate of overflows is 40,87% meaning that more than one in three containers exceed their capacity. On the other hand, the vehicle's average utilization is relatively high at 73,45%.

Experiment with extended container capacity

After an extension of container capacities by one third and simultaneously reducing the number of weekly collection routes from three to two which will be performed on Monday and Thursday now, the evaluation of the static routing strategy is reperformed with the same dataset used in the previous simulation.

Table 4-2: Simulation results of static routing strategy with extended container capacities – infeasible solution

Date	Weight (kg)	Distance (m)	Collected bins	Overf. bins (Si≥100%)	Overf. ratio (%)	Vehicle Utilization (%)
Thursday, 2 August 2018	154,00	316 127,24	18	0	0,00	7,70
Monday, 6 August 2018	2584,74	311 660,25	17	8	44,44	129,24
Thursday, 9 August 2018	1898,51	426 932,02	18	2	11,11	94,93
Monday, 13 August 2018	2680,83	330 680,75	18	10	55,56	134,04
Thursday, 16 August 2018	1698,93	354 370,29	17	2	11,11	84,95
Monday, 20 August 2018	2864,40	373 436,37	18	12	66,67	143,22
Thursday, 23 August 2018	1673,06	296 115,54	17	0	0,00	83,65
Monday, 27 August 2018	2566,26	417 117,09	18	8	44,44	128,31
Thursday, 30 August 2018	1728,50	360 813,08	18	1	5,56	86,42
Total Sum	17 849,22	3 187 252,63		Average	26,54	99,16

The results show a total travel distance of 3.187.252,63m meaning a decrease of approximately 31% compared to the first simulation with smaller containers. Additionally, the usage of bigger containers leads to a reduced rate of container overflows which is now 26% meaning only one in four containers exceeds its fill level on average. When looking at the vehicle utilization more closely, it shows that the static routing strategy will lead to an infeasible solution since the vehicle capacity is exceeded in several cases. As the vehicle exceeds almost every Monday its capacity, this is an indicator that the number of collection routes needs to be increased to handle the amount of waste between Thursday and Monday.

When increasing the number of weekly collection routes by one, the simulation results in a feasible solution:

Table 4-3: Simulation results of static routing strategy with extended container capacities - feasible solution

Date	Weight (kg)	Distance (m)	Collected bins	Overf. bins (Si≥100%)	Overf. ratio (%)	Vehicle Utilization (%)
Thursday, 2 August 2018	294,45	316 127,24	18	0	0,00	14,72
Saturday, 4 August 2018	1319,47	311 650,33	18	0	0,00	65,97
Monday, 6 August 2018	1344,11	311 660,25	17	0	0,00	67,21
Thursday, 9 August 2018	1898,51	426 932,02	18	2	11,11	94,93
Saturday, 11 August 2018	1447,60	420 956,44	18	0	0,00	72,38
Monday, 13 August 2018	1477,17	330 680,75	18	0	0,00	73,86
Thursday, 16 August 2018	1742,05	354 370,29	17	1	5,56	87,10
Saturday, 18 August 2018	1351,50	19 853,36	1	0	0,00	67,58
Monday, 20 August 2018	1610,22	346 466,04	17	1	5,56	80,51
Thursday, 23 August 2018	1680,45	296 115,54	17	0	0,00	84,02
Saturday, 25 August 2018	1267,73	331 771,79	17	0	0,00	63,39
Monday, 27 August 2018	1340,42	417 117,09	18	0	0,00	67,02
Thursday, 30 August 2018	1730,96	360 813,08	18	0	0,00	86,55
Total Sum		4 244 514,22		Average	1,71	71,17

With three collection routes per week, the total travel distance remains unchanged compared to the initial solution with smaller containers. The average overflow of containers has decreased tremendously to 1,71% and the vehicle utilization does not exceed 100% whereas this configuration results in a feasible solution.

Advanced Static Routing

When looking at the travel distances of each route containing all the 18 containers, it does not seem that route optimization heuristics are applied by the company. Therefore, this section aims to propose an optimized static routing strategy which should also be compared to the proposed two-stage stochastic optimization model ensuring that the comparison of the routing strategies is practical and critical.

Since the total travel distances vary greatly, ranging from 296 115,54m to 426 932,02m, a route optimization heuristic is applied to optimize the routes performed three times a week. As the implementation includes a Nearest Neighbor Algorithm in the second stage for the creation of initial routes, this algorithm is applied to the route of 18 containers and results in a distance of 175 715,61m. The collection days remain the same as in the initial static approach and will be

performed on every Monday, Thursday, and Saturday. This leads to 13 collection routes in August 2018 and results in a total travel distance of:

$$175\,715{,}61m * 13\ days = 2\,284\,302{,}93m$$

The rate of container overflows of 40,87% and the vehicle utilization of 73,45% remains the same.

Advanced Static Routing with extended container capacity

Also for the experiment with extended container capacity, an advanced static routing calculation will be performed. In this case, the amount of collection routes decreased to two routes per week, leading to a total of nine routes in August 2018.

$$175\,715{,}61m * 9\ days = 1\,581\,440{,}49m$$

The rate of container overflows of 26,54% and the vehicle utilization of 99,16% remains the same.

Since the simulation revealed that two collection routes per week will result in an infeasible solution, the feasible optimized total travel distance will stay the same with 13 collection routes scheduled in August 2018 and a total travel distance of:

$$175\,715{,}61m * 13\ days = 2\,284\,302{,}93m$$

The rate of container overflows of 1,71% and the vehicle utilization of 57,77% remains the same.

Dynamic Routing Strategy

For the evaluation of the two-stage stochastic optimization model the same dataset from the static routing approach is applied to guarantee an identical simulation environment. This serves as an enabler to attribute differences solely to a routing strategy rather than external factors. As the proposed dynamic routing approach includes adjustable parameters like planning horizon, vehicle capacity and threshold for must-go containers, a parameter tuning is performed in the first step of the analysis to determine the most promising configuration.

For the parameter tuning, several simulations are performed with different threshold values, since the planning horizon has been studied during the analysis of the predictive model in chapter 4.4 and vehicle capacity is set to 2.000kg. To find the optimal threshold value a tradeoff between total travel distance and container overflow rate must be made, as minimizing total travel distance will lead to less routes and consequently result in more overflowing containers and vice versa. Additionally, also the vehicle utilization will be affected by the configuration of the container threshold, since more collection visits will lead to decreased waste collected during one route. Therefore, the decision maker must define its objectives initially and then adjust the configuration to suit individual needs. Since this study deals with medical waste collection, a robust system is required that avoids overflowing containers to reduce public health risks, which will be considered during the parameter tuning.

To optimize the configuration of the dynamic approach, the simulation is performed with three different container thresholds of 80%, 85%, and 90%, meaning that the container will be scheduled for collection when the fill level exceeds the corresponding threshold. The results are then analyzed in terms of the objectives of increasing cost-effectiveness and creating a robust approach with minimized container overflows.

Table 4-4: Results with container threshold of 80%

Date	Weight (kg)	Distance (m)	Collected bins	Overf. bins (Si≥100%)	Overf. ratio (%)	Vehicle Utilization (%)
01.08.2018	174,94	10 737,91	1,00	1,00	5,56	8,75
02.08.2018	383,15	148 517,17	5,00	-	-	19,16
03.08.2018	1 254,18	80 805,56	10,00	6,00	33,33	62,71
04.08.2018	437,36	93 319,14	4,00	1,00	5,56	21,87
05.08.2018	529,76	99 979,20	8,00	1,00	5,56	26,49
06.08.2018	566,72	127 488,00	5,00	1,00	5,56	28,34
07.08.2018	1 434,05	175 715,61	18,00	4,00	22,22	71,70
08.08.2018	-	-	-	-	-	-
09.08.2018	1 027,49	175 715,61	18,00	-	-	51,37
10.08.2018	-	-	-	-	-	-
11.08.2018	1 406,94	175 715,61	18,00	-	-	70,35
12.08.2018	960,96	175 715,61	18,00	-	-	48,05
13.08.2018	-	-	-	-	-	-
14.08.2018	720,72	125 256,39	7,00	4,00	22,22	36,04
15.08.2018	508,82	70 797,11	4,00	2,00	11,11	25,44
16.08.2018	460,77	95 117,78	4,00	2,00	11,11	23,04
17.08.2018	894,43	139 380,19	8,00	3,00	16,67	44,72
18.08.2018	473,09	97 508,87	4,00	2,00	11,11	23,65
19.08.2018	979,44	133 994,45	8,00	3,00	16,67	48,97
20.08.2018	112,11	23 103,46	1,00	-	-	5,61
21.08.2018	1 783,94	175 715,61	18,00	5,00	27,78	89,20
22.08.2018	-	-	-	-	-	-
23.08.2018	229,15	58 038,97	3,00	-	-	11,46
24.08.2018	794,64	107 052,54	6,00	4,00	22,22	39,73
25.08.2018	613,54	75 038,86	5,00	2,00	11,11	30,68
26.08.2018	309,23	58 038,97	3,00	2,00	11,11	15,46
27.08.2018	399,17	79 739,70	3,00	2,00	11,11	19,96
28.08.2018	1 871,41	175 715,61	18,00	7,00	38,89	93,57
29.08.2018	-	-	-	-	-	-
30.08.2018	-	-	-	-	-	-
31.08.2018	1 923,15	175 715,61	18,00	4,00	22,22	96,16
		2 853 923,54			10,04	32,66

With a container threshold of 80%, the total travel distance is 2.853.923,54m for the whole of August 2018. This setting resulted in an average rate of container overflows of 10,04% and a vehicle utilization of 32,66%.

Table 4-5: Results with container threshold of 85%

Date	Weight (kg)	Distance (m)	Collected bins	Overf. bins (Si≥100%)	Overf. ratio (%)	Vehicle Utilization (%)
01.08.2018	171,25	10 737,91	1,00	1,00	5,56	8,56
02.08.2018	-	-	-	-	-	-
03.08.2018	2 035,26	175 715,61	18,00	5,00	27,78	101,76
04.08.2018	-	-	-	-	-	-
05.08.2018	1 363,82	175 715,61	18,00	1,00	5,56	68,19
06.08.2018	182,34	66 190,23	4,00	-	-	9,12
07.08.2018	1 288,67	175 715,61	18,00	1,00	5,56	64,43
08.08.2018	-	-	-	-	-	-
09.08.2018	391,78	109 061,64	4,00	-	-	19,59
10.08.2018	1 448,83	138 851,09	14,00	4,00	22,22	72,44
11.08.2018	-	-	-	-	-	-
12.08.2018	412,72	102 134,46	4,00	1,00	5,56	20,64
13.08.2018	665,28	60 038,80	5,00	4,00	22,22	33,26
14.08.2018	1 310,85	134 183,64	9,00	8,00	44,44	65,54
15.08.2018	123,20	40 228,85	1,00	-	-	6,16
16.08.2018	1 305,92	175 715,61	18,00	3,00	16,67	65,30
17.08.2018	-	-	-	-	-	-
18.08.2018	1 264,03	168 616,47	17,00	-	-	63,20
19.08.2018	532,22	86 665,30	5,00	-	-	26,61
20.08.2018	-	-	-	-	-	-
21.08.2018	1 623,78	135 060,03	13,00	8,00	44,44	81,19
22.08.2018	-	-	-	-	-	-
23.08.2018	1 193,81	175 715,61	18,00	2,00	11,11	59,69
24.08.2018	128,13	71 469,75	2,00	-	-	6,41
25.08.2018	227,92	61 524,59	2,00	-	-	11,40
26.08.2018	1 628,70	152 978,50	16,00	6,00	33,33	81,44
27.08.2018	-	-	-	-	-	-
28.08.2018	655,42	115 149,55	6,00	3,00	16,67	32,77
29.08.2018	797,10	67 889,15	7,00	2,00	11,11	39,86
30.08.2018	309,23	67 465,57	3,00	-	-	15,46
31.08.2018	1 084,16	140 964,13	9,00	4,00	22,22	54,21
		2 607 787,71			9,50	32,49

A container threshold of 85% results in a total travel distance of 2.607.787,71m, an average rate of container overflows of 9,50%, and an average vehicle utilization of 32,49% for the applied dataset from August 2018.

Table 4-6: Results with container threshold of 90%

Date	Weight (kg)	Distance (m)	Collected bins	Overf. bins (Si≥100%)	Overf. ratio (%)	Vehicle Utilization (%)
01.08.2018	171,25	10 737,91	1,00	1,00	5,56	8,56
02.08.2018	-	-	-	-	-	-
03.08.2018	1 183,95	141 747,44	9,00	4,00	22,22	59,20
04.08.2018	799,57	109 151,11	7,00	3,00	16,67	39,98
05.08.2018	596,29	110 380,58	5,00	2,00	11,11	29,81
06.08.2018	344,96	70 615,92	3,00	-	-	17,25
07.08.2018	1 271,42	117 747,75	10,00	5,00	27,78	63,57
08.08.2018	528,53	118 162,54	5,00	2,00	11,11	26,43
09.08.2018	473,09	86 096,06	5,00	1,00	5,56	23,65
10.08.2018	704,70	91 395,83	5,00	4,00	22,22	35,24
11.08.2018	308,00	82 946,70	2,00	2,00	11,11	15,40
12.08.2018	1 859,09	145 884,88	16,00	7,00	38,89	92,95
13.08.2018	-	-	-	-	-	-
14.08.2018	1 556,02	175 715,61	18,00	5,00	27,78	77,80
15.08.2018	-	-	-	-	-	-
16.08.2018	-	-	-	-	-	-
17.08.2018	554,40	74 197,69	5,00	2,00	11,11	27,72
18.08.2018	922,77	134 731,03	8,00	5,00	27,78	46,14
19.08.2018	874,72	106 035,53	5,00	5,00	27,78	43,74
20.08.2018	423,81	60 919,57	3,00	2,00	11,11	21,19
21.08.2018	1 349,04	172 159,97	12,00	6,00	33,33	67,45
22.08.2018	243,94	71 825,09	2,00	-	-	12,20
23.08.2018	1 023,79	161 954,60	17,00	2,00	11,11	51,19
24.08.2018	-	-	-	-	-	-
25.08.2018	264,88	75 737,45	3,00	-	-	13,24
26.08.2018	1 181,49	105 958,53	9,00	7,00	38,89	59,07
27.08.2018	289,52	65 044,72	2,00	2,00	11,11	14,48
28.08.2018	1 670,59	175 715,61	18,00	5,00	27,78	83,53
29.08.2018	-	-	-	-	-	-
30.08.2018	-	-	-	-	-	-
31.08.2018	1 155,62	109 990,09	9,00	4,00	22,22	57,78
		2 574 852,21			13,62	31,86

The container threshold of 90% results in total travel distance of 2.574.852,21m, an average rate of container overflows of 13,62%, and an average vehicle utilization of 31,86% for the applied dataset from August 2018.

The parameter analysis allows to conclude that the two-stage stochastic optimization model behaves as initially assumed, since total travel distance and the rate of container overflows are indirectly correlated. The analysis clearly shows that a lower container threshold leads to earlier scheduled collection visits which provides some space for forecasting errors, if actual arrival rates are higher than anticipated by the model and therefore ensures a more robust configuration. Consequently, the container threshold acts as a buffer for forecasting errors of the predictive model and directly affects the rate of container overflows and the total travel distance. The vehicle utilization is dependent on the number of visited containers and its fill levels. The more waste gets collected during one route, the higher the vehicle utilization. Therefore, vehicle utilization is not a reliable metric to determine the efficiency since visiting all containers lead to a high utilization but also includes visits of containers with potentially low fill levels resulting in unnecessary mileage.

Based on the results obtained from the simulation, a container threshold of 85% provides a good tradeoff between total travel distance and rate of overflowing containers. Compared to the results of 80%, a threshold of 85% leads to a decrease of 8,62% of total travel distance while maintaining the rate of overflows on the same level. The configuration with 90% only leads to a further decreased travel distance of 1,28% while increasing the rate of overflows by 3%.

The key finding of this parameter analysis highlights that a container threshold of 85% is well suited for the applied dataset and requirements for medical waste collection.

Experiment with extended container capacity

To examine the robustness and flexibility of the proposed dynamic approach, another simulation is performed with the same dataset but extended container capacity by one third as previously done for the static approach. Since increased container capacities mean relatively lower arrival rates, this simulation aims to examine how the dynamic approach behaves in situations with lower waste arrival rates as the model was faced with increased waste disposal rates like in epidemics during previous simulations. Therefore, this procedure allows to draw conclusions about model's ability to adapt to changing environments.

Table 4-7: Results of dynamic routing with extended container capacity

Date	Weight (kg)	Distance (m)	Collected bins	Overf. bins (Si≥100%)	Overf. ratio (%)	Vehicle Utilization (%)
01.08.2018	107,18	10 737,91	1,00	-	-	5,36
02.08.2018	-	-	-	-	-	-
03.08.2018	1 096,48	107 312,20	9,00	3,00	16,67	54,82
04.08.2018	293,22	86 872,06	3,00	1,00	5,56	14,66
05.08.2018	370,83	62 251,43	4,00	1,00	5,56	18,54
06.08.2018	453,38	56 854,43	4,00	1,00	5,56	22,67
07.08.2018	773,70	114 746,58	8,00	1,00	5,56	38,68
08.08.2018	325,25	92 633,93	3,00	-	-	16,26
09.08.2018	220,53	50 893,85	2,00	-	-	11,03
10.08.2018	672,67	118 879,67	6,00	1,00	5,56	33,63
11.08.2018	186,03	45 603,49	2,00	-	-	9,30
12.08.2018	676,37	78 341,97	5,00	4,00	22,22	33,82
13.08.2018	1 106,34	175 715,61	18,00	-	-	55,32
14.08.2018	-	-	-	-	-	-
15.08.2018	-	-	-	-	-	-
16.08.2018	-	-	-	-	-	-
17.08.2018	1 659,50	173 733,34	17,00	4,00	22,22	82,98
18.08.2018	123,20	23 535,31	2,00	-	-	6,16
19.08.2018	192,19	64 342,27	2,00	-	-	9,61
20.08.2018	479,25	93 342,61	5,00	-	-	23,96
21.08.2018	468,16	59 475,45	4,00	2,00	11,11	23,41
22.08.2018	273,50	53 494,71	3,00	1,00	5,56	13,68
23.08.2018	413,95	89 845,02	5,00	2,00	11,11	20,70
24.08.2018	595,06	100 730,24	5,00	1,00	5,56	29,75
25.08.2018	331,41	84 557,41	3,00	-	-	16,57
26.08.2018	1 154,38	157 801,32	16,00	1,00	5,56	57,72
27.08.2018	-	-	-	-	-	-
28.08.2018	-	-	-	-	-	-
29.08.2018	156,46	54 645,71	2,00	1,00	5,56	7,82
30.08.2018	694,85	87 964,25	6,00	2,00	11,11	34,74
31.08.2018	756,45	101 856,71	6,00	3,00	16,67	37,82
		2 146 167,48			5,20	21,90

An extension of container capacities by one third results in total travel distance of 2.146.167,48m, an average rate of container overflows of 5,2%, and an average vehicle utilization of 21,90% for the applied dataset from August 2018.

Comparative Analysis of Small-Scale Static and Dynamic Routing Strategies

After simulating static and dynamic routing strategies with the same dataset of small-scale instances, following key observations could be identified:

- Total travel distance: When using the static routing strategy with collection routes on three days a week, the total travel distance for August 2018 is calculated to be 4.617.950,59m, whereas the proposed dynamic approach results in a total travel distance of 2 607 787,71m during the same period. Therefore, the proposed dynamic approach shows a decrease in 44% of total travel distance for the applied dataset and consequently an increased cost-effectiveness.

  When comparing the dynamic approach with the advanced static routing approach, an increase of 14% can be observed, but at the same time, the rate of container overflows decreased by 31%, which is an important aspect since the approach will be used for medical waste collection, where overflowing containers pose potential health risks to society.
- Rate of container overflows: The rate of container overflows for static routing is relatively high at 40% on average, meaning almost every second container exceeds its capacity per period. When applying the same dataset on the proposed dynamic approach, the average rate of overflows is 9,5%, showing that the proposed model achieves a tremendous decrease in container overflows and therefore meets the initial objective to avoid overflows aiming to create a robust decision support system with high service level quality.

  Since the dataset is related to undifferentiated waste with relatively high arrival rates, the proposed model was able to prove its robustness during periods with increased amount of waste like in epidemics.
- Vehicle utilization: The vehicle utilization when using a static routing approach is quite high at 73%, whereas the utilization of the dynamic routing approach is much lower at 32%. This is caused by the initial decision during the parameter tuning since the proposed model should provide a robust system for medical waste collection while avoiding container overflows. Due to this decision, containers are only visited when they exceed their threshold resulting in avoiding unnecessary mileage and in less waste being carried during one route compared to the static approach.

- Robustness: The results of the experiments with extended container capacities showed that the proposed dynamic approach is flexible and adapts to changing environments, whereas the static approach resulted in an infeasible solution in the first run.
  An increase of container capacities by 33% resulted in an unchanged total travel distance for the static routing strategy. The dynamic approach leads to a total travel distance of 2.146.167,48m meaning a decrease by 18% compared to dynamic routing with normal container capacities and a decrease of 6% compared to the static approach with extended container capacities. The average rate of container overflows is very low in both cases, 1,71% for the static and 5,2% for dynamic strategy which is a difference of 3,5%.

The results of the small-scale experiments show that the proposed dynamic routing approach leads to increased cost-effectiveness by ensuring robustness, since the model reacts to changes in waste arrival rates and delivers feasible, cost-effective and robust solutions throughout the simulation which was not the case for the static routing strategy.

4.5.3. Large-Scale Experiments

The performance of large-scale experiments aims to strengthen the results of the small-scale experiments and additionally examine the performance of the proposed dynamic approach for larger instances in terms of scalability and practicability. The dataset used for experiments is published by Spinelli et al. (2024) and provided by a Portuguese waste collection company.

As outlined in chapter 514.3.2, the dataset includes the sensor data of 121 containers for plastic and metal waste and is split into six intervals: 0%, 12.5%, 37.5%, 62.5%, 87.5%, and Overflow. For the experiments, 50 containers are randomly selected and include the fill levels from 20 visits between April and July 2019. During simulation, it is assumed that the dataset represents fill levels of consecutive days, since there is no date of collection visits available. In this case it is assumed that each container has a maximum fill level of 240l which equals a weight of 123,2kg. Since a vehicle serves up to 50 containers per day, it must be able to carry the waste and therefore it is assumed that the vehicle has a capacity of 4.000kg of waste which equals a volume of 7.800l.

Static Routing Strategy

The assumption that static routing strategies perform container collections regardless of current fill levels on predefined days remains unchanged. In this case, all containers are visited 20 times during the period between April and July 2019. Therefore, also 20 fill level dates are available per container since they were gathered during the collection routes rather than using sensors.

To determine the total travel distance, the nearest neighbor algorithm is applied to a set of 50 containers to obtain a near optimal solution, as no information related to collection routes is provided. The nearest neighbor algorithm obtained a total travel distance of 134,66km for all 50 containers. During the time between April and July 2019 with 20 visits of each container, a total travel distance of 2.693,20km can be obtained:

$$134{,}66km * 20\ collection\ routes = 2693{,}20km$$

Table 4-8: Results of static routing strategy with 50 containers

Day	Weight (kg)	Distance (km)	Collected bins	Overf. bins (Si≥100%)	Overf. ratio (%)	Vehicle Utilization (%)
1	1 632,40	134,66	50,00	1,00	2,00	40,81
2	1 971,20	134,66	50,00	1,00	2,00	49,28
3	2 525,60	134,66	50,00	-	-	63,14
4	2 494,80	134,66	50,00	-	-	62,37
5	1 570,80	134,66	50,00	1,00	2,00	39,27
6	677,60	134,66	50,00	-	-	16,94
7	1 663,20	134,66	50,00	2,00	4,00	41,58
8	1 878,80	134,66	50,00	2,00	4,00	46,97
9	1 139,60	134,66	50,00	1,00	2,00	28,49
10	2 032,80	134,66	50,00	-	-	50,82
11	1 632,40	134,66	50,00	2,00	4,00	40,81
12	1 570,80	134,66	50,00	1,00	2,00	39,27
13	1 909,60	134,66	50,00	-	-	47,74
14	1 755,60	134,66	50,00	-	-	43,89
15	2 156,00	134,66	50,00	2,00	4,00	53,90
16	1 447,60	134,66	50,00	2,00	4,00	36,19
17	2 217,60	134,66	50,00	1,00	2,00	55,44
18	3 603,60	134,66	50,00	10,00	20,00	90,09
19	2 217,60	134,66	50,00	-	-	55,44
20	2 156,00	134,66	50,00	-	-	53,90
		2 693,20			2,60	47,82

The evaluation of the provided container fill levels showed an average rate of container overflows of 2,6% and an average vehicle utilization per route of 47,82%.

Dynamic Routing Strategy

To evaluate the performance of the proposed dynamic approach an extended dataset including 50 containers instead of 18 compared to the small-scale experiments is applied to the algorithm. Again, the dataset is the same as used for the large-scale evaluation of the static routing strategy to ensure an identical simulation environment.

Table 4-9: Results of static routing strategy with 50 containers

Day	Weight (kg)	Distance (km)	Collected bins	Overf. bins (Si≥100%)	Overf. ratio (%)	Vehicle Utilization (%)
1	-	-	-	-	-	-
2	500,50	89,84	5,00	2,00	4,00	12,51
3	1 640,10	93,93	13,00	5,00	10,00	41,00
4	2 408,05	102,09	19,00	7,00	14,00	60,20
5	503,07	65,45	4,00	2,00	4,00	12,58
6	337,26	73,81	3,00	-	-	8,43
7	1 359,82	77,07	11,00	4,00	8,00	34,00
8	1 097,51	72,32	8,00	6,00	12,00	27,44
9	740,23	74,88	6,00	2,00	4,00	18,51
10	819,28	91,29	6,00	3,00	6,00	20,48
11	1 607,25	89,23	13,00	3,00	6,00	40,18
12	854,19	67,73	7,00	1,00	2,00	21,35
13	831,60	67,27	6,00	4,00	8,00	20,79
14	1 324,40	90,59	11,00	1,00	2,00	33,11
15	1 093,40	76,46	9,00	1,00	2,00	27,34
16	954,80	88,17	8,00	2,00	4,00	23,87
17	1 478,40	99,61	11,00	5,00	10,00	36,96
18	3 619,00	116,82	27,00	11,00	22,00	90,48
19	554,40	74,08	6,00	-	-	13,86
20	1 262,80	78,05	11,00	-	-	31,57
		1 588,69			5,90	28,73

When applying the dataset of 50 containers to the proposed dynamic approach, it results in a total travel distance of 1.588,69km, an average rate of container overflows of 5,90% and an average vehicle utilization of 28,73%.

Comparative Analysis of Static and Dynamic Routing Strategies

After simulating static and dynamic routing strategies with the same dataset of 50 containers, following key observations could be identified:

- Total travel distance: Using a dynamic routing strategy for the applied dataset leads to a decrease from 2.693,20km to 1.588,69km meaning a reduction by 41%. Also when comparing the number of visited containers of both approaches, the corresponding Figure 4-13 shows that the dynamic approach results in a drastically decreased number of container visits compared to the static routing strategy.

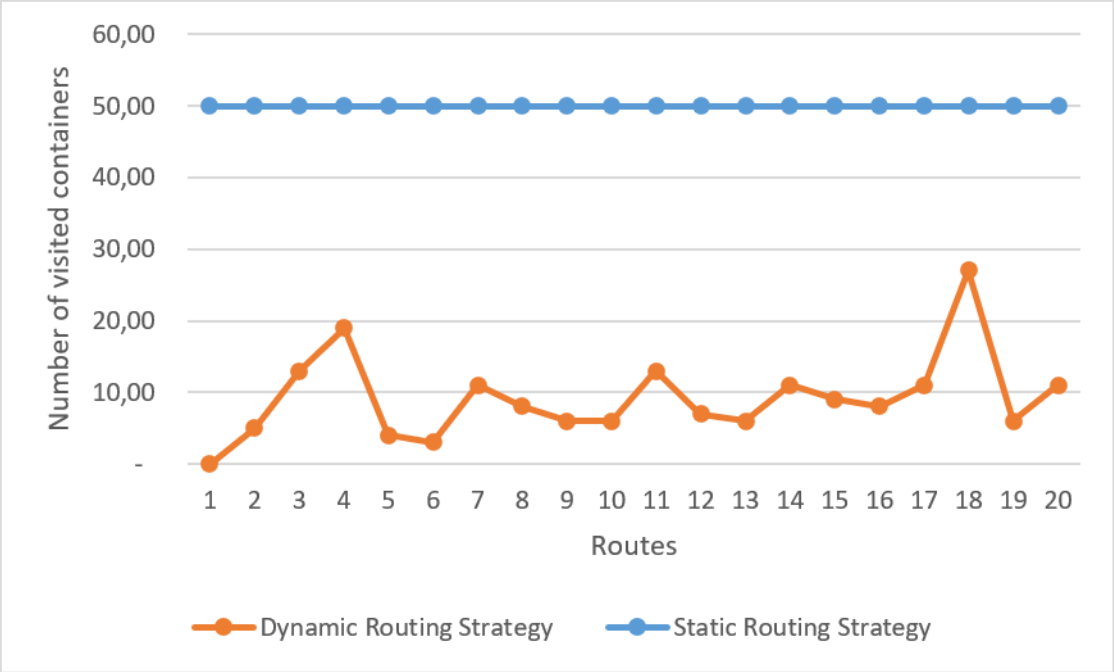

Figure 4-13: Comparative analysis of collection visits from both strategies

Compared to the results of the small-scale experiments, the obtained result provides a tremendous improvement, indicating a good performance especially for larger problems.

- Rate of container overflows: The average rate of container overflows increased from 2,60% for the static approach to 5,90% for the dynamic approach. Nevertheless, if compared the increase of 3,30% with the simultaneous reduction of total travel distance by 41%, this is still a substantial improvement where the increased rate of container overflows may not be significant.

  The rate of container overflows at 5,90% is nearly identical to that from the small-scale experiment for dynamic strategy with extended container capacities at 5,20%.

- Vehicle utilization: The large-scale experiment has proven that the assumptions from small-scale experiments were correct, that a reduction of total travel distance and rate of container overflows, also result in a decreased vehicle utilization, since the utilization for the static approach is calculated with 47,82% whereas the dynamic approach achieves 28,73%.
- Flexibility and Robustness: Since the dynamic approach achieves a reduction by 41% in total travel distance while keeping the rate of container overflows on a low level of 5,90%, the large-scale experiments explored that the proposed two-stage stochastic optimization model is able to handle also 50 containers which demonstrates that the proposed approach can adapt to changing environments while maintaining optimized decision making.

The performance of large-scale experiments demonstrate the model's scalability and practicability, since the proposed two-stage stochastic optimization model decreases the total travel distance by 41% for the applied dataset. The evaluation also reveals that the effect of the dynamic approach in terms of reduced total travel distance is particularly noticeable in cases with lower waste arrival rates, whereas in periods with higher arrival rates the focus is more on avoiding the rate of container overflows, resulting in a robust and efficient decision-making tool for medical waste collection.

4.6. Simulation-Optimization Approach

This section aims to demonstrate the connection between simulation and optimization and how the results of the simulation are fed into the mathematical model to obtain an exact solution. The process of this simulation-optimization approach starts with receiving the current container fill levels which are sent to the decision maker. The fill levels then serve as an input for the predictive model which is part of the simulation to determine forecasts based on a combination of historical and current fill levels. The output from the simulation are forecasts for each container which are fed into the mathematical model in the next step. The mathematical model then determines, based on the predicted fill levels, those containers that are expected to overflow soon and schedules them for collection. Finally, it calculates the shortest path to visit all containers that need to be visited. The process of the simulation-optimization approach is illustrated in Figure 4-14.

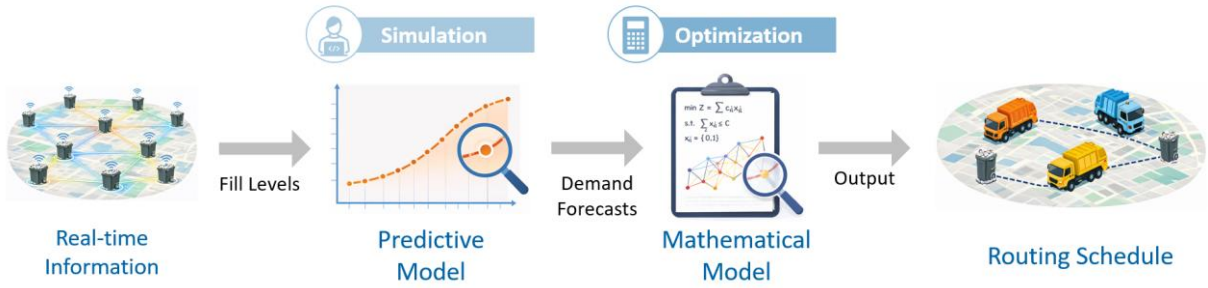

Figure 4-14: Process of simulation-optimization approach

Therefore, a framework on how the parameters from simulation are fed into the mathematical model is first introduced aiming to provide a better understanding of how simulation and optimization is connected. To ensure that the results from simulation are acceptable, data validation with real-world data is performed. Afterwards, numerous parameters from the simulation approach are used as input parameters in the mathematical model to solve the dynamic waste collection problem by using an exact solver.

4.6.1. Framework

The Framework ensures that the data from simulation is fed into the mathematical model aiming to solve the dynamic waste collection problem. Therefore, this framework outlines how the data from simulation is mapped into the mathematical model to ensure an identical environment.

Mapping

The mathematical model needs numerous input parameters to calculate the exact solution of the given problem. Below, all input parameters applied to the mathematical model are described and how they can be mapped to parameters in simulation experiments:

- **Distance Matrix:** The distance matrix remains unchanged compared to the simulation. The mathematical model uses the same distance matrix used in the small- and large-scale experiments with 18 and 50 containers and serves as a basis to determine the distances between node i and j represented by d_{ij} in the mathematical model.
- **Datasets:** The datasets which are used as input data for arrival rates are identical to the simulation. There are three scenarios, two for small-scale and one for large-scale instances. The small-scale instances contain real-world data from 18 containers, and an adapted scenario with decreased arrival rates to simulate the behavior in periods with

lower waste accumulation. The large-scale instance contains real-world arrival rates of 50 containers aiming to simulate a more realistic and practical scenario.

- **Forecasted arrival rates:** To address the stochastic nature of waste accumulation, the expected arrival rates for each container i are determined by the predictive model which is part of the simulation. The forecasts serve as input for expected daily accumulation rate a_i of each container i.
- **Fill Levels:** Fill levels S_i are calculated by cumulating the daily arrival rates, whereas the daily arrival rates are taken from the real-world datasets for small- and large-scale experiments which are distributed according to a normal distribution. To avoid container overflows, the expected arrival rates a_i from the predictive model are added to the current fill levels S_i to obtain the predicted fill levels F_i for each container i.
- **Critical Threshold for Must-Go Containers:** The threshold used in the mathematical model is 85%, since parameter tuning performed during simulation in chapter 4.5.2 has shown that this value provides a promising tradeoff between minimizing total travel distance and container overflows which are inversely proportional to each other.
- **Container Capacity:** Also the container capacity E_i remains unchanged compared to the simulation of small- and large-scale experiments, ensuring that the collection visits are triggered at the same point of filling volume ψ.
- **Fleet:** The fleet size is initialized with one vehicle, and the same vehicle capacity Q as used in the small- and large-scale experiments during simulation. This ensures that the capacity constraints are identical to the simulation. The vehicles for small-scale instances have a capacity of 2.000kg whereas the vehicle capacity for large-scale instances is 4.000kg.
- **Costs:** The costs per kilometer are set to one, since the costs are directly correlated to the total travel distance. Therefore, the objective value that should be minimized is the total travel distance aiming to draw conclusions about the results of total travel distances obtained in the simulation in terms of solution quality.

The information flow into the mathematical model is illustrated by Figure 4-15.

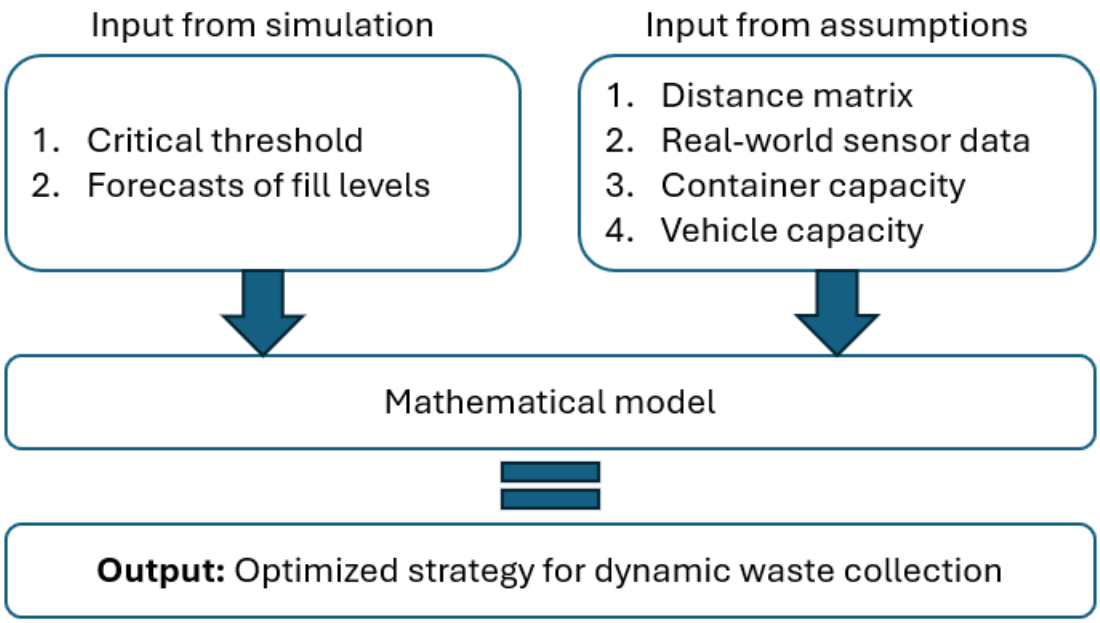

Figure 4-15: Data input into the mathematical model

Since the mathematical model determines the optimal route for each day based on expected fill levels, the total travel distance for the considered period will be determined by cumulating the daily solutions.

4.6.2. Validation of input data from simulation

This section aims to draw conclusions about the accuracy of the estimated stochastic waste accumulation rates. Therefore, the estimates from simulation are compared to the real-world data to prove the claim that the simulation provides highly accurate forecasts. The analysis starts by assessing the accuracy of forecasts for each scenario and ends with an assessment of all forecasts.

Data Validation for small-scale instance with normal arrival rates

To validate the accuracy of forecasts from simulation, the confidence interval with 95% of simulation and real-world data is plotted in Figure 4-16.

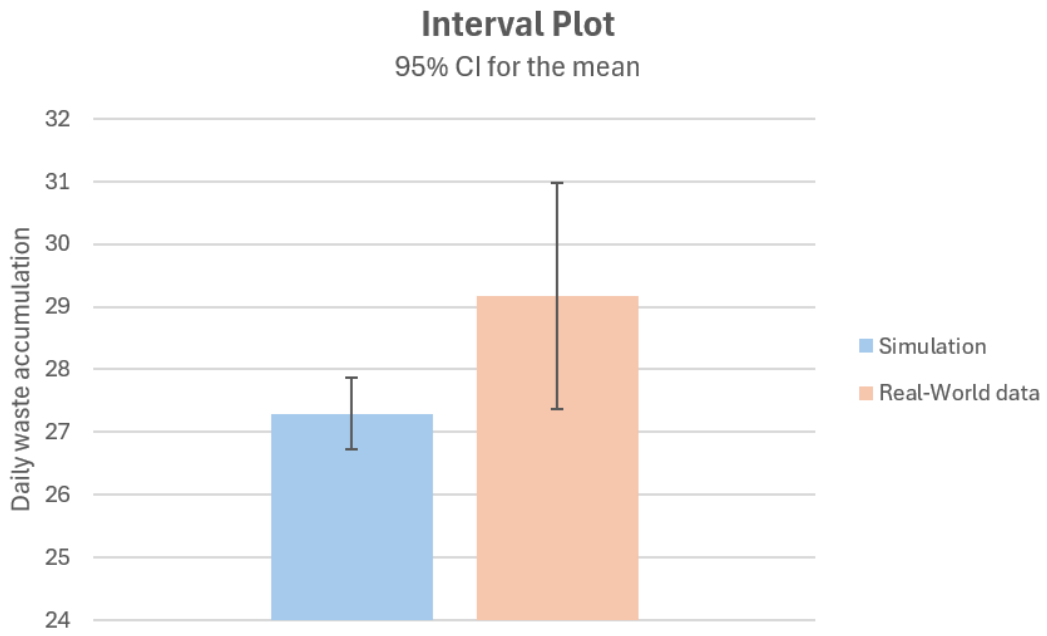

Figure 4-16: Confidence interval (CI) for small-scale instance with normal arrival rate

The result shows that the forecasts do not provide sufficient accuracy, since the confidence intervals are not sufficiently overlapped. To increase the accuracy the key findings from section 4.4.2 can be leveraged by using test data to train the predictive model resulting in increased accuracy. Since the real-world data from de Morais et al. (2024) provides data from several months, fill level data starting from 15th July 2018 was used to train the predictive model. Afterwards, the forecasts for August 2018 were determined again based on the trained predictive model. The results obtained are shown in Figure 4-17.

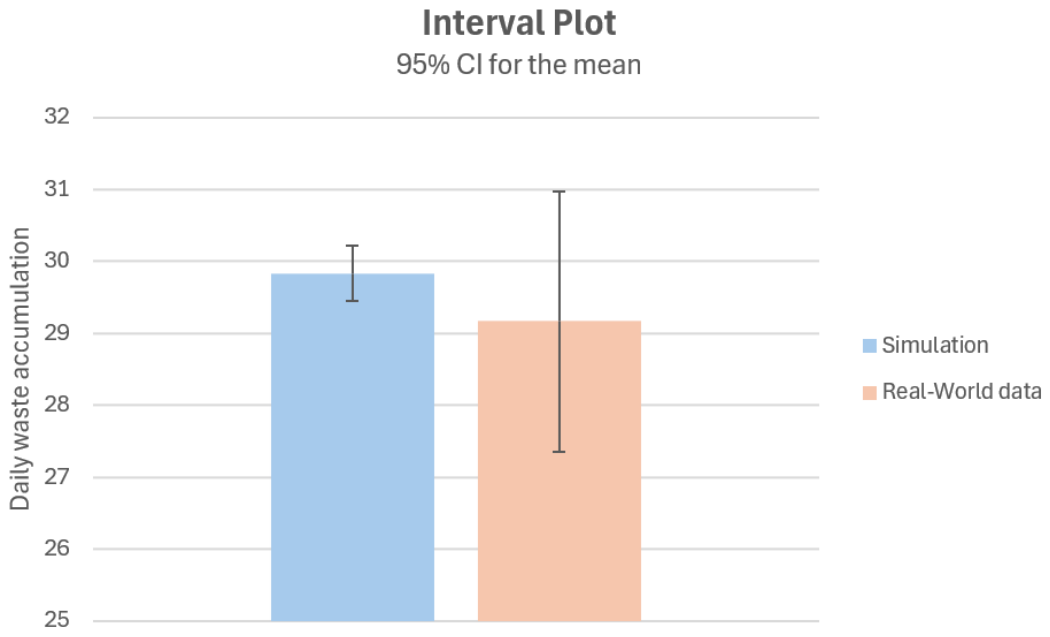

Figure 4-17: Confidence interval (CI) for small-scale instance with normal arrival rate after test data

The results after applying test data to the predictive model indicate an increased accuracy, since the confidence intervals of simulation and real-world data are strongly overlapping, strengthening the assumption from section 4.4.2, that the predictive model learns over time and therefore increases accuracy of forecasts.

Data Validation for small-scale instance with reduced arrival rates

Compared to the assessment with normal arrival rates, the accuracy of forecasts for small instance with reduced arrival rates indicates accurate forecasts even without testing data, as illustrated in Figure 4-18, since the confidence intervals are overlapping.

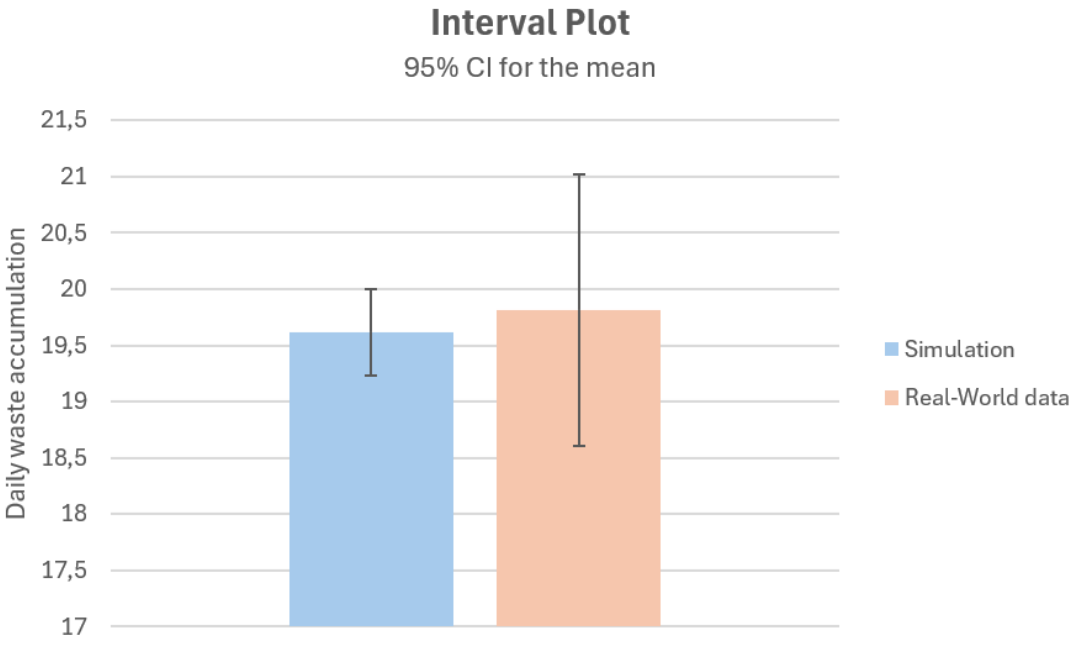

Figure 4-18: Confidence interval (CI) for small-scale instance with normal arrival rate

Afterwards, a set with training data was applied for reduced arrival rates consisting of real-world data between 15th and 31st of July 2018 published by de Morais et al. (2024). Then, the forecasts for August 2018 were determined again by the predictive model and compared to the real-world data by using confidence intervals in Figure 4-19. Again, the result shows good solution quality of the forecasts obtained by simulation, since the confidence intervals are overlapping.

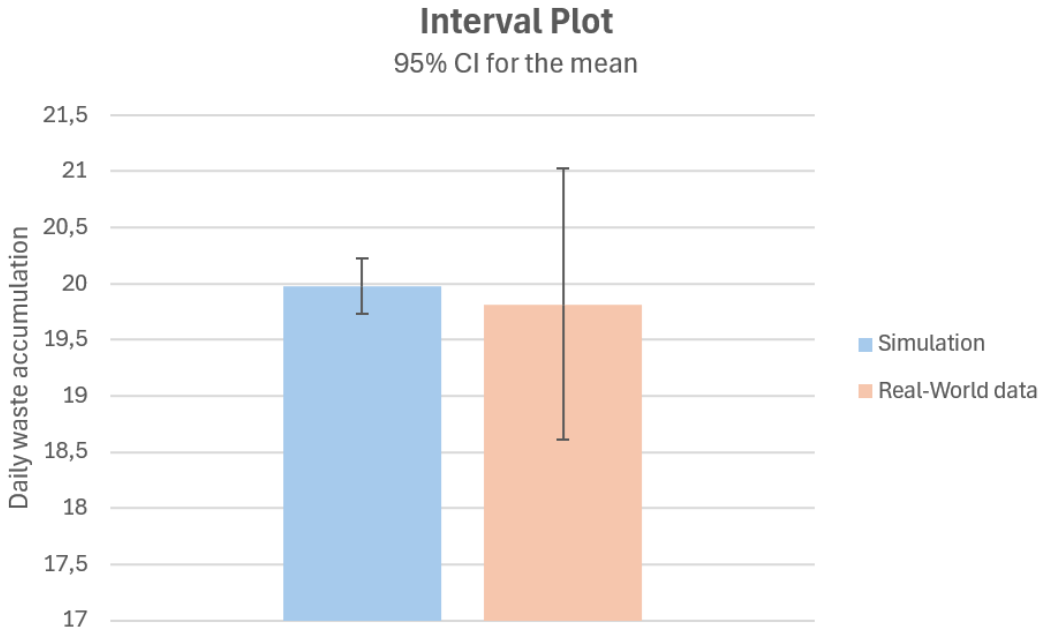

Figure 4-19: Confidence interval (CI) for small-scale instance with reduced arrival rate after test data

Data Validation for large-scale instance

Also for the large instance, consisting of 50 containers, a separate assessment regarding the accuracy of the forecasts obtained from simulation is performed aiming to examine the solution quality for different problem sizes. But in this scenario, the fact that fill levels are classified in five intervals rather than exact integer values, poses a limitation, as it is not possible to determine with certainty whether the forecasts are correct. The confidence intervals of forecasts from simulation and real-world data are shown in Figure 4-20, indicating acceptable accuracy, as the confidence intervals are strongly overlapping.

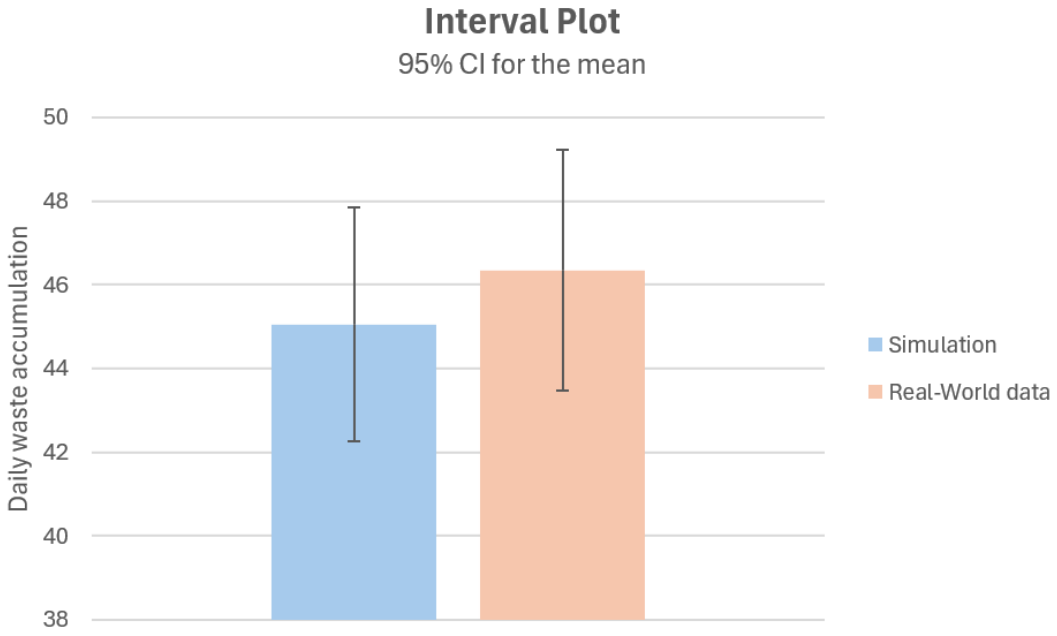

Figure 4-20: Confidence interval (CI) for large-scale instance

Data Validation for all instances

Assessing the accuracy for each scenario indicated promising results, also for the large instance despite the limitation of accuracy due to fill level intervals rather than exact values. Therefore, an assessment of all forecasts is performed to draw conclusion about the accuracy of the predictive model in general. The results are illustrated in Figure 4-21, indicating accurate results, as the confidence intervals of simulation and real-world data are overlapping.

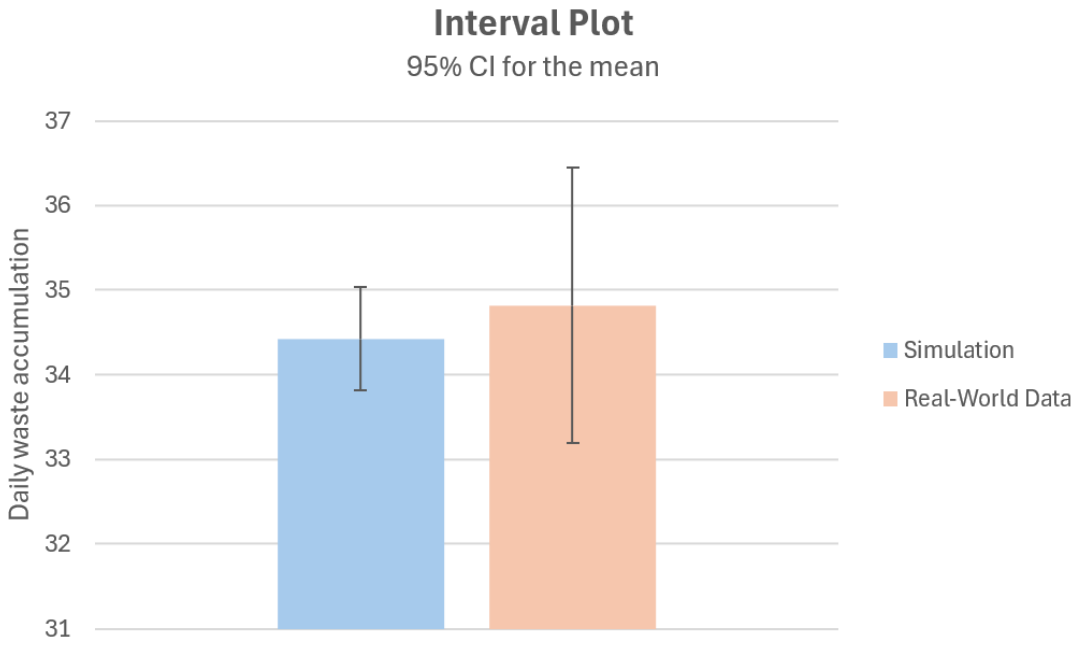

Figure 4-21: Confidence interval (CI) for all instances

To strengthen the results obtained from the graphical comparison of the confidence intervals, a two-sample T-Test with a significance level of 5% is performed aiming to determine whether there is a significant difference between forecasts of simulation and real-world data. If the p-value of the t-test is smaller or equal the defined significance level of 5%, the null-hypothesis can be rejected, meaning there is a significant difference between the two groups.

Null-Hypothesis (H_0): The means of forecasts from simulation and real-world data are equal and consequently there is no significant difference between these two groups.

Alternative-Hypothesis (H_a): The means of forecasts from simulation and real-world data are not equal, meaning that there is a significant difference.

The T-Test obtains a p-value of 0.66, meaning that the null hypothesis cannot be rejected and concludes that there is no significant difference between the forecasts from simulation and real-world data. Consequently, with 95% confidence interval, the accuracy of the predictive model can be classified as properly and acceptably.

4.6.3. Results of the Mathematical Model

Since the accuracy of the forecasts determined by the predictive model from simulation is at an adequate accurate level, they can now be used as an input for the optimization part to solve the given problem exactly. To ensure a detailed analysis of several scenarios, the mathematical model was solved for small-scale and large-scale datasets resulting in three scenarios aiming to examine the model's behavior under different waste accumulation rates and problem sizes.

Scenario 1: Small-Scale Instance with real-world data

To obtain the exact results for scenario 1 from the mathematical model, the same parameters are fed into the model, as previously described in the framework, and are visualized in Table 4-10.

Table 4-10: Results of simulation-optimization for scenario 1

Period of collection	Results from mathematical model		
	Distance [km]	Visited containers	Rate of overflows [%]
First week	401,79	28	5,56
Second week	761,47	52	3,97
Third week	678,94	49	6,35
Fourth week	676,59	46	3,97
Fifth week	448,01	33	3,33
Total	2 966,79	208	4,66

The results show that the exact solution results in a total travel distance of 2 966,79km, visits in total 208 containers and leads to rate of container overflows of 4,66%. The results indicate that the forecasts from simulation provide adequate accurate forecasts since only 4,66% containers exceed their maximum capacity.

Scenario 2: Small-Scale Instance with adapted arrival rates

As in scenario 1 the same input parameters from the simulation are fed into the mathematical model to ensure identical environments. The results obtained for scenario 2 are shown in Table 4-11.

Table 4-11: Results of simulation-optimization for scenario 2

Period of collection	Results from mathematical model		
	Distance [km]	Visited containers	Rate of overflows [%]
First week	369,44	20	3,33
Second week	559,52	34	0,79
Third week	569,27	35	0,79
Fourth week	523,09	29	1,59
Fifth week	428,68	23	2,22
Total	2 450,00	141	1,61

For scenario 2, the results show a similar pattern to the first scenario. The solution of the mathematical model leads to a total travel distance of 2 450,00km, visits 141 containers in total and only 1,61% of containers exceed their maximum capacity. Also in this scenario, the low number of overflowing containers is an indicator that the forecasts from simulation are highly accurate.

Scenario 3: Large-Scale Instance

Also in scenario 3, the input parameters from the simulation of large instances are fed into the mathematical model according to the established framework. The results of the exact solution are shown in Table 4-12.

Table 4-12: Results of simulation-optimization for scenario 3

Period of collection	Results from mathematical model		
	Distance [km]	Visited containers	Rate of overflows [%]
First week	619,72	72	2,57
Second week	545,29	70	1,43
Third week	578,43	81	3,33
Total	1 743,44	223	2,40

The results of the mathematical model for 50 containers result in a total travel distance of 1 743,44km and 223 container visits in total over one month. With 2,40%, the rate of overflows shows a similar result as in the small-scale instances and again indicating a good solution quality of forecasts received from simulation.

Discussion of results of the exact solution

Since the rate of container overflows obtained from the mathematical model ranging between 1,61% and 4,66%, this further supports the key findings from data validation in terms of solution quality and accuracy of forecasts. The low number of container overflows demonstrates that the anticipated container fill levels are accurate enough such that the rate of

container overflows remains under 5%, even in scenarios with high demands, resulting in a resilient and robust system.

4.6.4. Comparison of results from simulation-optimization and static routing

In this section, the results from the simulation-optimization approach are compared to the results from static routing to evaluate the performance. Therefore, a performance comparison is done for three scenarios. The most important aspects for the comparison are the total travel distance and the rate of container overflows, since they allow to draw conclusions about cost-effectiveness and robustness. The comparison of the total travel distance is shown in Figure 4-22 and the rate of container overflows is illustrated in Figure 4-23.

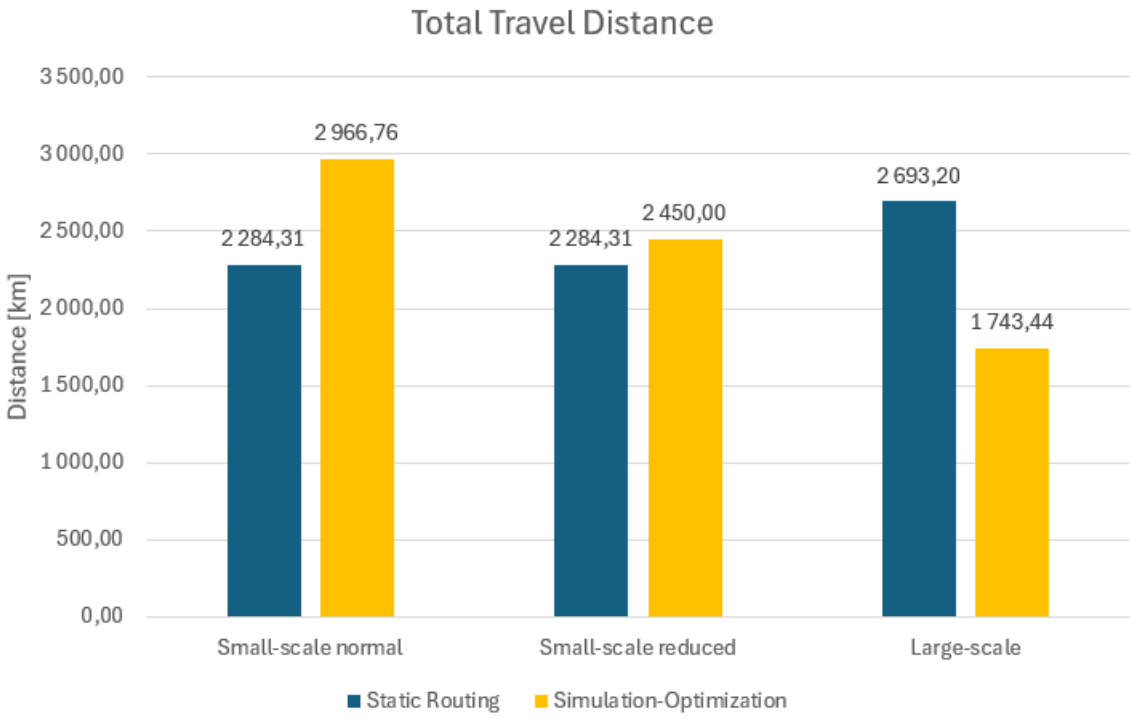

Figure 4-22: Comparison of total travel distance between static routing and simulation-optimization approach

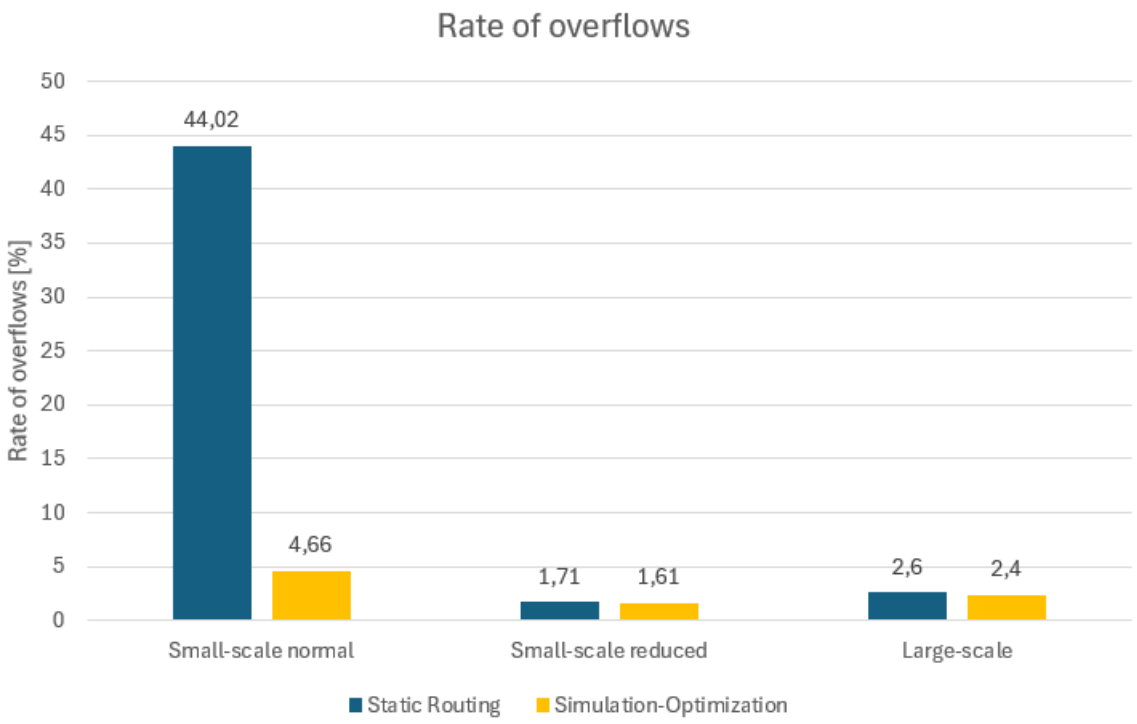

Figure 4-23: Comparison of rate of container overflows between static routing and simulation-optimization approach

The results indicate that the total travel distance slightly increases when using the simulation-optimization approach for both scenarios of the small-scale instances, but the rate of container overflows, especially at the scenario with higher arrival rates tremendously decreases from 44,02% to 4,66% when using a dynamic routing approach. For the large-scale instance the dynamic routing approach decreased the total travel distance by 35% compared to the static routing approach. The rate of container overflows slightly decreased from 2,6% at the static routing strategy to 2,4% at the dynamic routing strategy.

The findings of the comparison indicate that the dynamic routing approach is able to keep the rate of container overflows on a very low level demonstrating the flexibility of the dynamic routing approach since it is able to adapt to changing environments. Furthermore, the proposed approach also considers the total travel distance, since it highly impacts the cost-effectiveness leading to balanced solution of reducing total travel distance and avoiding overflowing containers.

4.6.5. Sensitivity Analysis

To examine the influence of changing environments on the performance of the proposed model, a sensitivity analysis is performed by changing the arrival rates on both the small- and large-

scale instances. Additionally, the results of the sensitivity analysis should also allow to draw conclusions about the practicability of the dynamic routing strategy and its behavior in different seasons, since the production of medical waste tends to be seasonal with increased waste production rates for example during influenza waves.

Since the arrival rates for small-scale instances originate from sensor data of undifferentiated waste, which is characterized by high arrival rates, the sensitivity analysis for small-scale instances examines cases with decreased demand. The sensor data for large-scale instances originates from plastic and metal waste, which is characterized by lower demands compared to undifferentiated waste. Therefore, the waste arrival rates will be both increased and decreased to examine the behavior in numerous scenarios for larger problem sizes.

The results of the mathematical model with increased and decreased waste arrival rates based on the datasets with 18 and 50 containers are presented and discussed in this chapter.

Small-scale instance with decrease of 33%

The results for decreased waste arrival rates by 33% for small-scale instance consisting of 18 waste containers are presented in Table 4-13.

Table 4-13: Results for small-scale instance with decreased arrival rates of 33%

Period of collection	Results from mathematical model			Change distance [%]	Change rate of overflows [%]
	Distance [km]	Visited containers	Rate of overflows [%]		
First week	369,44	20	3,33	-8,05	-40,00
Second week	559,52	34	0,79	-26,52	-80,00
Third week	569,27	35	0,79	-16,15	-87,50
Fourth week	523,09	29	1,59	-22,69	-60,00
Fifth week	428,68	23	2,22	-4,31	-16,67
Total	**2 450,00**	**141**	**1,61**	**-17,42**	**-65,38**

The results show that a decrease of 33% in waste arrival rates also lead to decreased total travel distance by 17,42% and rate of arrival rates by 65,38% resulting in an average rate of overflowing containers of 1,61%. In conclusion, the rates of change indicate that a change in waste arrival rates has relatively a greater impact on the rate of container overflows.

Small-scale instance with decrease of 50%

The results for decreased waste arrival rates by 50% for small-scale instance are presented in Table 4-14.

Table 4-14: Results for small-scale instance with decreased arrival rates of 50%

Period of collection	Results from mathematical model			Change distance [%]	Change rate of overflows [%]
	Distance [km]	Visited containers	Rate of overflows [%]		
First week	417,83	18	0,80	+3,99	-85,60
Second week	513,19	25	0,00	-32,61	-100,00
Third week	407,95	23	0,57	-39,91	-91,00
Fourth week	470,62	19	0,00	-30,44	-100,00
Fifth week	334,81	17	0,00	-25,27	-100,00
Total	**2 144,41**	**102**	**0,26**	**-27,71**	**-94,46**

A decrease of 50% in waste arrival rates lead to a decrease in total travel distance by 27,71% and improves the rate of overflows by 94,46% meaning that only 0,26% of waste containers exceed their capacity. Also in this scenario, the same pattern can be identified, since the change of waste arrival rates has relatively a greater impact on the rate of container overflows than on the total travel distance.

Large-scale instance with increase of 10%

The results for increased waste arrival rates of 10% for a large-scale instance consisting of 50 waste containers are presented in Table 4-15.

Table 4-15: Results for large-scale instance with increased arrival rates of 10%

Period of collection	Results from mathematical model			Change distance [%]	Change rate of overflows [%]
	Distance [km]	Visited containers	Rate of overflows [%]		
First week	678,26	97	6,00	+9,45	+133,33
Second week	583,82	82	3,14	+7,07	+120,00
Third week	591,48	88	5,00	+2,26	+50,00
Total	**1 853,56**	**267**	**4,70**	**+6,32**	**+95,83**

An increase of 10% in waste arrival rates lead to an increase of 6,32% in total travel distance and 95,83% in rate of container overflows. Also in the large-scale experiment, the rate of container overflows reacts disproportionately strongly to the changed waste arrival rates.

Large-scale instance with decrease of 25%

The results for decreased waste arrival rates of 25% for large-scale instance are presented in Table 4-16.

Table 4-16: Results for large-scale instance with decreased arrival rates of 25%

Period of collection	Results from mathematical model			Change distance [%]	Change rate of overflows [%]
	Distance [km]	Visited containers	Rate of overflows [%]		
First week	552,20	64	3,71	-10,90	+44,44
Second week	532,21	55	0,29	-2,40	-80,00
Third week	455,12	57	5,00	-21,32	+50,00
Total	**1539,53**	**176**	**2,90**	**-11,70**	**+20,83**

A decrease of 25% in waste arrival rates led to a decrease of 11,70% in total travel distance and to an increase of 20,83% in rate of container overflows. In this scenario, a decrease in waste arrival rates leads to an increase in container overflows from 2,4% to 2,9%, meaning a small change on a very low level.

Discussion of results from sensitivity analysis

The sensitivity analysis highlights the impact of changes in waste arrival rates and shows that lower demand leads to less travel distance and less container overflows. The key findings are that in relative terms, the total travel distance is less sensitive than the rate of container overflows. It can also be observed that the dynamic routing strategy performs well in numerous scenarios, always resulting in a rate of container overflows below 5%. Consequently, based on the results of the sensitivity analysis it can be concluded that the dynamic routing strategy provides a resilient and flexible approach which is capable to adapt to changing environments and therefore provides an efficient tool for decision making.

5. Conclusion

5.1. Introduction

This chapter aims to summarize the findings from the comparative analysis of the static and dynamic routing approaches to solve the underlying dynamic and stochastic reverse IRP for medical waste collection focusing on the aspect of cost-effectiveness while maintaining robustness in terms of avoiding container overflows.

The dynamic and stochastic reverse IRP incorporates several challenges for medical waste collectors, especially due to the dual requirement of minimizing the inversely proportional variables total travel distance and rate of container overflows. A significant challenge poses the uncertain nature of waste arrivals, making it difficult to determine collection schedules which serve as a base for efficient route planning and consequently affect the total travel distance. This research aimed to propose a two-stage stochastic optimization model that uses heuristics to solve the problem also for larger instances to overcome computational limitations of exact solvers. By examining the model's performance compared to traditional static routing strategies based on real-world data, this study provides insights into the development and implementation of a dynamic routing strategy based on real-time sensor data followed by an assessment of its scalability and practicability aiming to support companies in decision-making.

This chapter first highlights the obtained answers to the research questions defined in chapter 1.2, followed by a conclusion to summarize findings. Based on these findings, managerial insights are presented. Afterwards, limitations of the study and future work are discussed.

5.2. Answers to the Main Research Question

To answer the main research question: "How to enhance cost-effectiveness of medical waste collection by using real-time data provided by sensors?", a two-stage stochastic optimization model was developed, implemented, and tested to examine the performance compared to traditional static routing strategies which are used from companies that provided real-world fill level data. The proposed model consists of an integrated predictive model to address the stochastic nature of waste arrival rates, which allows to anticipate future container fill levels and schedules collection routes accordingly to avoid containers from exceeding their capacity limits. To optimize collection routes, an algorithm was implemented which first creates initial tours and then optimizes them by applying a savings algorithm aiming to merge tours to reduce total travel distance.

As the algorithm is based on anticipated fill levels, the accuracy of forecasts is key for the efficiency of the dynamic routing approach, since inaccurate forecasts lead to inefficient container visits and consequently increases total travel distance and negatively affects the overall operational efficiency. Therefore, the performed validation of forecasts by using confidence interval plots and a two-sample T-Test demonstrated that the integrated predictive model determines forecasts, with confidence interval of 95%, properly and acceptably. This key finding serves as a basis for efficient dynamic routing, both for the mathematical model and the heuristic approach. Furthermore, the results of the mathematical model demonstrated the accuracy of the forecasts from the predictive model, since the rate of overflowing containers were relatively low ranging from 1,66% to 4,66% during higher demands, resulting in a resilient and robust routing strategy.

Additionally, the performance of small- and large-scale experiments allowed an in-depth evaluation of the performance and behavior under several different circumstances to examine cost-effectiveness, robustness, scalability, and practicability of the proposed dynamic approach compared to traditional static routing. The results show that especially for large-scale experiments a tremendous increase of cost-effectiveness can be achieved which proves the model's ability to adapt to changing environments as well as its scalability.

5.3. Conclusion

This study examined the implementation of a two-stage stochastic optimization approach to solve the dynamic and stochastic reverse IRP for medical waste collection to address present challenges like cost-effectiveness and avoiding overflowing containers to reduce health risks to society and being better prepared for future pandemics. Therefore, the experiments for both the simulation and the simulation-optimization aim to determine the ability of the proposed approach to address the existing challenges of medical waste management as well as its practicability.

Cost-effectiveness: Since cost-effectiveness increased especially in cases with smaller arrival rates, with a reduced total travel distance by 35% at large-scale instance, the analysis shows that the proposed approach is well suited for those cases in terms of cost-effectiveness.

Robustness: The experiments show that the proposed dynamic approach provides a robust solution with low rate of container overflows in both cases of high and low arrival rates.

Especially in cases with high arrival rates as observed during small-scale experiments with normal container capacities, the dynamic approach ensures a low rate of container overflows of 4,66%, whereas the static approach results in 44,02%. Consequently, the proposed approach maintains a high service level quality especially in scenarios with higher arrival rates contributing to improve robustness.

Scalability: By performing experiments with different scenarios and problem sizes, the proposed model demonstrated its adaptability by providing feasible solutions, even in case of an extension of container capacity, where the static routing strategy resulted in an infeasible solution. Especially the large-scale experiments explored the model's scalability, since the total travel distance decreased by 35% compared to the static approach for the applied real-world dataset.

Key takeaways: In conclusion, the proposed two-stage stochastic optimization model has demonstrated that it is very well suited to solve the dynamic and stochastic reverse IRP and to address present challenges of medical waste collection like cost-effectiveness and avoiding container overflows. Especially when comparing the proposed approach to the traditional static approach, the large-scale experiment shows a tremendous reduction of total travel distance by 35%. Also in scenarios with high waste arrival rates, which is the case during epidemics and pandemics, the dynamic approach showed good performance in decreasing the rate of container overflows to 4,66% compared to 44,02% at the static routing strategy. Therefore, the findings of the experiments reveal an increased cost-effectiveness while maintaining a low rate of container overflows, resulting in an efficient and robust system for the applied datasets.

5.4. Managerial Insights

The key findings of this study offer valuable insights for managers in industry dealing with waste collection management in general and using static routing strategies. The dynamic routing strategy in combination with a predictive model showed promising results to anticipate future waste accumulation rates and therefore reduces overflowing containers while increasing service level quality. Consequently, the combination of demand forecasting and route optimization provides an efficient decision-support tool for waste management companies aiming to optimize cost-effectiveness and service level quality.

Another strength of the implemented heuristic is scalability, since the exact solver is not capable to provide exact solutions within reasonable amount of time for larger problem instances whereas the results from simulation of the heuristic approach demonstrate promising solution quality also for larger problems ensuring practicability to tackle real-world challenges.

Additionally, since the dynamic routing approach uses real-time data, it ensures that decisions are adequately adapted to current situations aiming to increase flexibility while ensuring rapid decision making which is crucial to improve operational efficiency.

5.5. Limitations and future work

The initial assumptions and circumstances, such as limited availability of real-world sensor data especially for medical waste entails four limitations which offer future research areas:

First, the predictive model is based on a simple procedure following a normal distribution. To further improve the proposed approach, a detailed study regarding an optimal forecasting method for this topic can be performed to obtain more accurate forecasts resulting in improved route planning. Also, the aspect of using AI to anticipate fill levels can be an interesting topic for future developments.

Second, the size of datasets used for simulation is limited to a maximum of 50 containers, since there is no real-world data published including more than 50 containers and a corresponding distance matrix. Even though the dataset from Spinelli et al. (2024) entailed sensor data from 121 containers, the distance matrix included only 50 containers. Therefore, future work can analyze the behavior of the dynamic routing approach on enlarged datasets, to provide insights into the performance of more realistic waste management systems.

Third, since the dataset from Spinelli et al. (2024) is related to plastic and metal waste, and the dataset from de Morais et al. (2024) related to undifferentiated waste, real-world sensor data for medical waste is missing. Especially undifferentiated waste has high arrival rates which is leveraged during small-scale experiments to examine the behavior of the proposed dynamic model in scenarios like epidemics with increased waste arrival rates. Future work can fill this gap by analyzing the behavior of the model for real-world sensor data for medical waste to strengthen the findings from this thesis.

Fourth, it is assumed that each container is equipped with a sensor, since the number of hospitals and pharmacies is relatively low compared to traditional municipal waste collection. Therefore, future work can perform a cost-benefit analysis to determine if all containers need to be equipped with sensors to improve cost-effectiveness and robustness of medical waste collection.

References

Abaku, E. A., Edunjobi, T. E., & Odimarha, A. C. (2024). Theoretical approaches to AI in supply chain optimization: Pathways to efficiency and resilience. *International Journal of Science and Technology Research Archive*, *6*(1), 092–107. https://doi.org/10.53771/ijstra.2024.6.1.0033

Alarcon Ortega, E. J., & Doerner, K. F. (2023). A sampling-based matheuristic for the continuous-time stochastic inventory routing problem with time-windows. *Computers & Operations Research*, *152*, 106129. https://doi.org/10.1016/j.cor.2022.106129

Alarcon Ortega, E. J., Malicki, S., Doerner, K. F., & Minner, S. (2024). Stochastic inventory routing with dynamic demands and intra-day depletion. *Computers & Operations Research*, *163*, 106503. https://doi.org/10.1016/j.cor.2023.106503

Alsalibi, B., Babaeianjelodar, M., & Venkat, I. (2012). A Comparative Study between the Nearest Neighbor and Genetic Algorithms: A revisit to the Traveling Salesman Problem. *International Journal of Computer Science and Electronics Engineering*, *1*, 34–38.

Bard, J. F., Huang, L., Jaillet, P., & Dror, M. (1998). A Decomposition Approach to the Inventory Routing Problem with Satellite Facilities. *Transportation Science*, *32*(2), 189–203. https://doi.org/10.1287/trsc.32.2.189

Bell, W. J., Dalberto, L. M., Fisher, M. L., Greenfield, A. J., Jaikumar, R., Kedia, P., Mack, R. G., & Prutzman, P. J. (1983). Improving the Distribution of Industrial Gases with an On-Line Computerized Routing and Scheduling Optimizer. *Interfaces (Providence)*, *13*(6), 4–23. https://doi.org/10.1287/inte.13.6.4

Campbell, A. M., & Savelsbergh, M. W. P. (2004). A Decomposition Approach for the Inventory-Routing Problem. *Transportation Science*, *38*(4), 488–502. https://doi.org/10.1287/trsc.1030.0054

Charaf, S., Taş, D., Flapper, S. D. P., & Van Woensel, T. (2024). A matheuristic for the two-echelon inventory-routing problem. *Computers & Operations Research*, *171*, 106778. https://doi.org/10.1016/j.cor.2024.106778

Coelho, L. C., Cordeau, J.-F., & Laporte, G. (2014). Heuristics for dynamic and stochastic inventory-routing. *Computers & Operations Research*, *52*(Part A), 55–67. https://doi.org/10.1016/j.cor.2014.07.001

Cubillos, M., Spliet ,Remy, & and Wøhlk, S. (2022). On the effect of using sensors and dynamic forecasts in inventory-routing problems. *INFOR: Information Systems and Operational Research*, *60*(4), 473–490. https://doi.org/10.1080/03155986.2022.2073110

de Morais, C. S., Ramos, T. R. P., Lopes, M., & Barbosa-Póvoa, A. P. (2024). A data-driven optimization approach to plan smart waste collection operations. *International Transactions in Operational Research*, *31*(4), 2178–2208. https://doi.org/10.1111/itor.13235

Divakaran, S., Bethanney Janney, J., Charan, M., Krishnakumar, S., Mathangi, B., & Dhanalakshmi, K. (2023). A Smart Bin for Disposal of Infectious Medical Waste. *2023 International Conference on Artificial Intelligence and Knowledge Discovery in Concurrent Engineering (ICECONF)*, 1–5. https://doi.org/10.1109/ICECONF57129.2023.10083671

Elbek, M., Crainic, T. G., & Rei, W. (2015). *Multi-period collection of recyclable materials in a multi-compartment vehicle under uncertainty*.

Feng, Y., Che, A., & Tian, N. (2024). Robust inventory routing problem under uncertain demand and risk-averse criterion. *Omega*, *127*, 103082. https://doi.org/10.1016/j.omega.2024.103082

Fu, J., Pan, S., Liao, J., & Zhang, S. (2025). Application of Hybrid Optimization Algorithm Considering Yard Priority to Port Vehicle Scheduling. *2025 8th World Conference on Computing and Communication Technologies (WCCCT)*, 452–457. https://doi.org/10.1109/WCCCT65447.2025.11027996

Ghasemi, P., Goli, A., Goodarzian, F., & Ehmke, J. F. (2025). Simulation-based genetic algorithm for optimizing a municipal cooperative waste supply chain in a pandemic. *Engineering Applications of Artificial Intelligence*, *139*, 109478. https://doi.org/10.1016/j.engappai.2024.109478

Goodarzian, F., Ghasemi, P., Gunasekaran, A., & Labib, A. (2024). A fuzzy sustainable model for COVID-19 medical waste supply chain network. *Fuzzy Optimization and Decision Making*, *23*(1), 93–127. https://doi.org/10.1007/s10700-023-09412-8

Govindan, K., Nosrati-Abarghooee, S., Nasiri, M. M., & Jolai, F. (2022). Green reverse logistics network design for medical waste management: A circular economy transition through case approach. *Journal of Environmental Management*, *322*, 115888. https://doi.org/10.1016/j.jenvman.2022.115888

Gunawan, N., Rusdiansyah, A., & Isnaini, F. (2024). The Development of the Flexible Periodic Vehicle Routing Problem Model for Gas Cylinders Distribution. *2024 9th International Conference on Business and Industrial Research (ICBIR)*, 1054–1059. https://doi.org/10.1109/ICBIR61386.2024.10875709

Haseltalab, S., & Karimi, H. (2025). Enhancing efficiency in delivering petroleum products: Optimizing the deteriorating inventory-routing problem with heterogeneous multi-compartment vehicles. *International Journal of Management Science and Engineering Management*, *20*(2), 159–182. https://doi.org/10.1080/17509653.2024.2422308

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

Hussain, Dr. I., Elomri, Dr. A., Kerbache, Dr. L., & Omri, Dr. A. E. (2024). Smart city solutions: Comparative analysis of waste management models in IoT-enabled environments using multiagent simulation. *Sustainable Cities and Society*, *103*, 105247. https://doi.org/10.1016/j.scs.2024.105247

Ishaq, A., Mohammad, S. J., Bello, A.-A. D., Wada, S. A., Adebayo, A., & Jagun, Z. T. (2023). Smart waste bin monitoring using IoT for sustainable biomedical waste management. *Environmental Science and Pollution Research*. https://doi.org/10.1007/s11356-023-30240-1

Jorge, D., Pais Antunes, A., Rodrigues Pereira Ramos, T., & Barbosa-Póvoa, A. P. (2022). A hybrid metaheuristic for smart waste collection problems with workload concerns. *Computers & Operations Research*, *137*, 105518. https://doi.org/10.1016/j.cor.2021.105518

Ketzenberg, M. E., & Metters, R. D. (2020). Adapting operations to new information technology: A failed “internet of things” application. *Omega*, *92*, 102152. https://doi.org/10.1016/j.omega.2019.102152

Markov, I., Bierlaire, M., Cordeau, J.-F., Maknoon, Y., & Varone, S. (2020). Waste collection inventory routing with non-stationary stochastic demands. *Computers & Operations Research*, *113*, 104798. https://doi.org/10.1016/j.cor.2019.104798

Markov, I., Varone, S., & Bierlaire, M. (2016). Integrating a heterogeneous fixed fleet and a flexible assignment of destination depots in the waste collection VRP with intermediate facilities. *Transportation Research Part B: Methodological*, *84*, 256–273. https://doi.org/10.1016/j.trb.2015.12.004

Marković, D., Marković, S., Marinković, D., & Pamučar, D. (2024). DYNAMIC VEHICLE ROUTING PROBLEM FOR SMART WASTE COLLECTION. *Facta Universitatis, Series: Mechanical Engineering*, *0*. https://casopisi.junis.ni.ac.rs/index.php/FUMechEng/article/view/12598

Mes, M., Schutten, M., & Rivera, A. P. (2014). Inventory routing for dynamic waste collection. *Waste Management*, *34*(9), 1564–1576. https://doi.org/10.1016/j.wasman.2014.05.011

Mestouri, N., Kanoun, I., & Beji, N. (2024). GIS and Ant Colony Optimization for Hazardous Medical Waste Management in Tunisia. *2024 International Conference on Decision Aid Sciences and Applications (DASA)*, 1–5. https://doi.org/10.1109/DASA63652.2024.10836234

Nolz, P. C., Absi, N., & Feillet, D. (2014a). A Bi-Objective Inventory Routing Problem for Sustainable Waste Management Under Uncertainty. *Journal of Multi-Criteria Decision Analysis*, *21*(5–6), 299–314. https://doi.org/10.1002/mcda.1519

Nolz, P. C., Absi, N., & Feillet, D. (2014b). A stochastic inventory routing problem for infectious medical waste collection. *Networks*, *63*(1), 82–95. https://doi.org/10.1002/net.21523

Osaba, E., Yang, X.-S., Fister, I., Del Ser, J., Lopez-Garcia, P., & Vazquez-Pardavila, A. J. (2019). A Discrete and Improved Bat Algorithm for solving a medical goods distribution problem with pharmacological waste collection. *Swarm and Evolutionary Computation*, *44*, 273–286. https://doi.org/10.1016/j.swevo.2018.04.001

Pisinger, D., & Røpke, S. (2010). Large Neighborhood Search. In M. Gendreau (Ed.), *Handbook of Metaheuristics* (pp. 399–420). Springer.

Pulparambil, S., Al-Busaidi, A., Al-Hatimy, Y., & Al-Farsi, A. (2024). Internet of things-based smart medical waste management system. *Telematics and Informatics Reports*, *15*, 100161. https://doi.org/10.1016/j.teler.2024.100161

Rahmanifar, G., Mohammadi, M., Sherafat, A., Hajiaghaei-Keshteli, M., Fusco, G., & Colombaroni, C. (2023). Heuristic approaches to address vehicle routing problem in the Iot-based waste management system. *Expert Systems with Applications*, *220*, 119708. https://doi.org/10.1016/j.eswa.2023.119708

Rajakumari S T, & Roy, S. (2024). Waste Management: A Telegram — Integrated Smart Garbage Bin. *2024 2nd International Conference on Emerging Trends in Engineering and Medical Sciences (ICETEMS)*, 538–543. https://doi.org/10.1109/ICETEMS64039.2024.10964987

Ramos, T. R. P., de Morais, C. S., & Barbosa-Póvoa, A. P. (2018). The smart waste collection routing problem: Alternative operational management approaches. *Expert Systems with Applications*, *103*, 146–158. https://doi.org/10.1016/j.eswa.2018.03.001

Samyudha K, R, S. S., A, S., & Vincy R, F. (2025). Integrated IoT-based Hazard Detection and Automated Waste Management System for Environmental Safety. *2025 5th International Conference on Trends in Material Science and Inventive Materials (ICTMIM)*, 1175–1181. https://doi.org/10.1109/ICTMIM65579.2025.10987973

Saraswathi P, S., V, V. J., K Y, L., J V, P., M, P., & B, D. (2024). Optimizing Waste Management Systems: A Technological Approach. *2024 IEEE International Conference on Blockchain and Distributed Systems Security (ICBDS)*, 1–5. https://doi.org/10.1109/ICBDS61829.2024.10837381

Shaw, P. (1998). Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. In M. Maher & J.-F. Puget (Eds.), *Principles and Practice of Constraint Programming—CP98* (pp. 417–431). Springer. https://doi.org/10.1007/3-540-49481-2_30

Spinelli, A., Maggioni, F., Ramos, T. R. P., Barbosa-Póvoa, A. P., & Vigo, D. (2024). *A rolling horizon heuristic approach for a multi-stage stochastic waste collection problem* (No. arXiv:2405.14499; Version 1). arXiv. https://doi.org/10.48550/arXiv.2405.14499

SureshKumar, M., K., S. T., R, D., & S, S. (2025). Intelligent Waste Management: Automated Compression and Real-Time Fill-Level Monitoring. *2025 International Conference on Computing and Communication Technologies (ICCCT)*, 1–3. https://doi.org/10.1109/ICCCT63501.2025.11019889

Swapna, N. S., Rao, B. K., Netra, K., & Abhinav, P. (2025). IoT-Driven Intelligent Waste Management System with Disinfectant Spray. *2025 5th International Conference on Trends in Material Science and Inventive Materials (ICTMIM)*, 979–984. https://doi.org/10.1109/ICTMIM65579.2025.10987934

T, S., G, S., & T, R. D. (2024). Enhancing Waste Management Through Smart Bins. *2024 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS)*, 1–5. https://doi.org/10.1109/ICPECTS62210.2024.10780412

Tandon, R., & Bansal, A. (2023). IoT-enabled Smart Waste Management: Leveraging the Power of IoT for Sustainable Solutions. *2023 3rd International Conference on Innovative Sustainable Computational Technologies (CISCT)*, 1–6. https://doi.org/10.1109/CISCT57197.2023.10351210

Taslimi, M., Batta, R., & Kwon, C. (2020). Medical waste collection considering transportation and storage risk. *Computers & Operations Research*, *120*, 104966. https://doi.org/10.1016/j.cor.2020.104966

Thangam, S., M, G., Yadav, C. S. K., Harshith, M., & Kumar, M. S. (2024). A Smart Waste Management System Using ESP8266. *2024 4th International Conference on Sustainable Expert Systems (ICSES)*, 1–5. https://doi.org/10.1109/ICSES63445.2024.10762967

Trudeau, P., & Dror, M. (1992). Stochastic Inventory Routing: Route Design with Stockouts and Route Failures. *Transportation Science*, *26*(3), 171–184. https://doi.org/10.1287/trsc.26.3.171

Wu, H., Tao, F., & Yang, B. (2020). Optimization of Vehicle Routing for Waste Collection and Transportation. *International Journal of Environmental Research and Public Health*, *17*(14), Article 14. https://doi.org/10.3390/ijerph17144963

Yu, V. F., Salsabila, N. Y., Gunawan, A., & Siswanto, N. (2025). A three-stage matheuristic for the blood stochastic inventory routing problem. *Transportation Research Part E: Logistics and Transportation Review*, *200*, 104143. https://doi.org/10.1016/j.tre.2025.104143

Zhang, L. (2025). Optimization of Medical Waste Recycling Network Based on Variable Cycle Strategy. *2025 8th International Conference on Advanced Algorithms and Control Engineering (ICAACE)*, 1722–1725. https://doi.org/10.1109/ICAACE65325.2025.11019047

Zheng, M., Li, Y., Du, N., Wang, Q., Huang, E., & Jiang, P. (2024). Joint optimization of recyclable inventory routing problem under uncertainties in an incentive-based recycling system. *Computers & Industrial Engineering*, *198*, 110692. https://doi.org/10.1016/j.cie.2024.110692

Appendix

The Appendix provides the code for the mathematical model and the simulation part of the thesis. Furthermore, it also entails the datasets applied to both the simulation and simulation-optimization to ensure reproducibility.

Datasets

Distance matrix for small-scale instance experiments with 18 containers:

0,5493.04,11594.14,20439.78,35012.99,32700.8,35075.72,23121.35,20564.43,11476.86,11654.46,15064.18,25348.95,20609.49,24214.9,31081.9,20175.23,15245.29,10030.21
5244.87,0,6573.18,15418.82,29992.03,27679.85,30054.76,18100.39,14920.61,6455.9,6633.5,10043.23,20327.99,14965.68,19193.94,26060.94,14531.41,10224.34,5968.78
11388.68,6575.66,0,17427.69,32000.9,29688.72,32063.63,20109.26,19328.33,8464.77,991.14,4860.34,15145.11,19373.39,21202.81,28069.81,18939.12,5041.45,10065.67
19789.07,16261.61,17271.93,0,20484.82,18172.63,16536.02,4581.65,13792.91,13052.43,17332.26,20741.98,31026.74,17498.39,5675.2,12542.2,14616.03,20923.09,11980.62
34584.41,31056.95,32067.28,20146.58,0,6937.53,34782.52,15377.65,32039.41,27847.78,32127.6,35537.32,45822.09,38362.57,14747.37,14019.95,32862.53,35718.44,27018.07
31871.83,28344.37,29354.7,17434.0,6936.48,0,14337.19,12665.07,29326.83,25135.2,29415.02,32824.74,43109.5,35649.99,12034.79,8871.36,30149.95,33005.86,24305.49
34424.34,30896.88,31907.21,16514.0,35120.09,14310.36,0,17005.05,18255.57,27687.71,31967.53,35377.26,45662.02,21961.05,18098.6,8310.28,19078.69,35558.37,26615.9
22481.7,18954.24,19964.57,4571.36,14571.13,13029.77,17016.78,0,14273.67,15745.07,20024.89,23434.62,33719.38,17979.15,1385.31,6751.38,15096.79,23615.73,14673.25
18986.89,14890.82,18897.27,13782.62,32388.71,30076.53,18267.3,14273.67,0,14712.73,18957.59,16374.95,28213.57,1801.47,15367.22,14273.48,2145.57,16556.06,15739.46
23605.35,20077.89,21088.21,8963.36,23536.57,21224.39,23599.3,11644.93,20856.19,0,21148.54,24558.26,34843.02,27383.51,12738.48,19605.48,21679.31,24739.37,15250.32
11449.0,6635.98,991.14,17488.01,32061.22,29749.04,32123.95,20169.58,19388.65,8525.1,0,5851.48,16136.24,19433.71,21263.13,28130.13,18999.44,6032.59,10125.99
14858.73,10045.7,4860.34,20897.74,35470.95,33158.77,35533.68,23579.31,16367.68,11934.82,5851.48,0,10294.83,16412.74,24672.86,31539.86,15978.47,3972.17,13535.71
25143.49,20330.47,15145.11,31182.5,45755.71,43443.53,45818.44,33864.07,28213.57,22219.58,16136.24,10294.83,0,26697.5,34957.62,41824.62,26263.24,14256.93,23820.47
19031.96,14935.88,18942.33,17488.1,38293.92,35981.73,21972.78,17979.15,1801.47,14757.79,19002.65,16420.01,26704.78,0,19072.7,17978.96,4936.14,16601.12,15784.53
23575.25,20047.79,21058.12,5664.91,13940.86,12399.49,18110.33,1385.31,15367.22,16838.62,21118.44,24528.17,34812.93,19072.7,0,6121.1,16190.34,24709.28,15766.81
30442.25,26914.8,27925.12,12531.91,14018.9,8871.36,8322.01,6751.38,14273.48,23705.62,27985.44,31395.17,41679.93,17978.96,6121.1,0,15096.6,31576.28,22633.81
18597.69,14501.61,18508.06,14605.74,33211.83,30899.64,19090.42,15096.79,2145.57,14323.52,18568.39,15985.75,26270.51,4936.14,16190.34,15096.6,0,16166.86,15350.26
15039.84,10226.82,5041.45,21078.85,35652.06,33339.88,35714.79,23760.42,16548.79,12115.93,6032.59,3972.17,14256.93,16593.85,24853.97,31720.97,16159.58,0,13716.82
9823.15,5976.44,9950.35,12045.83,26855.13,24542.95,26681.78,14727.41,15757.34,7586.26,10010.67,13420.4,23705.16,15802.4,15820.96,22687.96,15368.13,13601.51,0

Distance matrix for large-scale experiments with 50 containers:

Arrival rates for small-scale experiments with 18 containers:

1 100 22 15 3 68 23 48 29 16 41 29 44 11 25 35 30 18 15 57 19 51 30 22 23 16 42 21 40 39 33 16
2 0 13 51 14 2 86 0 10 90 12 21 14 5 58 17 13 5 16 15 20 13 1 5 17 34 1 0 47 55
3 1 21 38 18 36 28 38 0 31 67 4 75 29 37 63 13 49 7 64 1 36 64 0 36 33 38 26 35 32 5 44
4 31 10 20 67 50 28 23 64 0 69 11 66 3 68 1 7 22 47 69 12 50 38 11 89 10 50 4 61 7 32 6
5 0 6 34 13 25 13 51 4 9 4 69 6 35 22 0 32 5 22 16 31 26 25 48 21 1 64 47 45 19 46 66
6 20 13 19 22 5 44 23 32 12 26 25 24 0 32 0 38 29 24 23 33 13 0 0 1 36 0 1 5 50 6 31
7 5 13 62 39 54 46 50 12 11 48 19 64 35 43 12 0 69 11 89 41 22 3 47 56 15 51 50 52 5 43 57
8 0 31 78 22 69 31 67 33 9 60 12 51 45 63 3 19 54 36 67 10 58 6 14 39 14 54 42 42 23 7 29
9 0 12 50 11 18 53 68 32 37 63 20 52 36 69 0 31 69 10 49 19 59 6 10 46 53 61 13 55 6 59 32
10 0 12 40 38 52 14 5 50 43 40 24 87 10 47 0 18 45 24 54 13 26 11 62 7 35 30 12 57 31 11 51
11 16 30 68 21 15 0 36 35 20 44 21 45 5 76 0 2 54 15 100 12 21 38 30 8 37 46 21 16 23 40 71
12 24 17 15 16 42 16 2 25 36 49 3 8 12 13 28 24 33 29 8 8 16 18 3 18 20 2 2 27 6 30 37
13 23 16 34 21 43 16 54 13 0 16 17 53 37 9 9 49 20 0 0 45 6 12 28 35 23 25 0 68 3 4 47
14 7 30 70 0 54 11 37 33 30 66 5 78 0 20 34 0 47 22 65 4 58 12 3 31 35 55 11 34 13 22 35
15 0 34 56 15 70 0 66 34 18 50 21 19 31 48 35 17 53 30 100 0 62 38 0 68 18 46 11 63 8 46 83
16 0 30 43 31 48 18 28 17 25 42 29 40 29 75 0 39 48 27 49 1 73 20 39 62 12 42 17 70 30 16 52
17 3 12 25 33 30 31 54 13 4 38 0 10 60 70 0 30 20 51 37 42 70 6 24 44 12 70 12 22 37 0 42
18 9 0 31 25 17 28 58 7 0 55 22 44 10 37 17 16 17 16 56 23 8 37 55 15 2 18 15 4 37 28 10

Arrival rates for small-scale experiments with 18 containers and reduced arrival rates by 33%:

1 67 15 10 2 46 16 32 20 11 28 20 30 8 17 24 20 12 10 38 13 34 20 15 16 11 28 14 27 26 22 11
2 0 9 34 10 2 58 0 7 60 8 14 10 4 39 12 9 4 11 10 14 9 1 4 12 23 1 0 32 37
3 1 14 26 12 24 19 26 0 21 45 3 50 20 25 42 9 33 5 43 1 24 43 0 24 22 26 18 24 22 4 30
4 21 7 14 45 34 19 16 43 0 46 8 44 2 46 1 5 15 32 46 8 34 26 8 60 7 34 3 41 5 22 44
5 0 4 23 9 17 9 34 3 6 3 46 4 24 15 0 22 4 15 11 21 18 17 32 14 1 43 32 30 13 31 44
6 14 9 13 15 4 30 16 22 8 18 17 16 0 22 0 26 20 16 16 22 9 0 0 1 24 0 1 4 34 4 21
7 4 9 42 26 36 31 34 8 8 32 13 43 24 29 8 0 46 8 60 28 15 2 32 38 10 34 34 35 4 29 38
8 0 21 52 15 46 21 45 22 6 40 8 34 30 42 2 13 36 24 45 7 39 4 10 26 10 36 28 28 16 5 20
9 0 8 34 8 12 36 46 22 25 42 14 35 24 46 0 21 46 7 33 13 40 4 7 31 36 41 9 37 4 40 22
10 0 8 27 26 35 10 4 34 29 27 16 58 7 32 0 12 30 16 36 9 18 8 42 5 24 20 8 38 21 8 34
11 11 20 46 14 10 0 24 24 14 30 14 30 4 51 0 2 36 10 67 8 14 26 20 6 25 31 14 11 16 27 48
12 16 12 10 11 28 11 2 17 24 33 2 6 8 9 19 16 22 20 6 6 11 12 2 12 14 2 2 18 4 20 25
13 16 11 23 14 29 11 36 9 0 11 12 36 25 6 6 33 14 0 0 30 4 8 19 24 16 17 0 46 2 3 32
14 5 20 47 0 36 8 25 22 20 44 4 52 0 14 23 0 32 15 44 3 39 8 2 21 24 37 8 23 9 15 24
15 0 23 38 10 47 0 44 23 12 34 14 13 21 32 24 12 36 20 67 0 42 26 0 46 12 31 8 42 6 31 56
16 0 20 29 21 32 12 19 12 17 28 20 27 20 50 0 26 32 18 33 1 49 14 26 42 8 28 12 47 20 11 35
17 2 8 17 22 20 21 36 9 3 26 0 7 40 47 0 20 14 34 25 28 47 4 16 30 8 47 8 15 25 0 28
18 6 0 21 17 12 19 39 5 0 37 15 30 7 25 12 11 12 11 38 16 6 25 37 10 2 12 10 3 25 19 7

Arrival rates for small-scale experiments with 18 containers and reduced arrival rates by 50%:

1 50 11 7 1 34 11 24 14 8 20 14 22 5 12 17 15 9 7 28 9 25 15 11 11 8 21 10 20 19 16 8
2 0 6 25 7 1 43 0 5 45 6 10 7 2 29 8 6 2 8 7 10 6 0 2 8 17 0 0 23 27 0 0
3 0 10 19 9 18 14 19 0 15 33 2 37 14 18 31 6 24 3 32 0 18 32 0 18 16 19 13 17 16 2 22
4 15 5 10 33 25 14 11 32 0 34 5 33 1 34 0 3 11 23 34 6 25 19 5 44 5 25 2 30 3 16 3
5 0 3 17 6 12 6 25 2 4 2 34 3 17 11 0 16 2 11 8 15 13 12 24 10 0 32 23 22 9 23 33
6 10 6 9 11 2 22 11 16 6 13 12 12 0 16 0 19 14 12 11 16 6 0 0 0 18 0 0 2 25 3 15
7 2 6 31 19 27 23 25 6 5 24 9 32 17 21 6 0 34 5 44 20 11 1 23 28 7 25 25 26 2 21 28
8 0 15 39 11 34 15 33 16 4 30 6 25 22 31 1 9 27 18 33 5 29 3 7 19 7 27 21 21 11 3 14
9 0 6 25 5 9 26 34 16 18 31 10 26 18 34 0 15 34 5 24 9 29 3 5 23 26 30 6 27 3 29 16
10 0 6 20 19 26 7 2 25 21 20 12 43 5 23 0 9 22 12 27 6 13 5 31 3 17 15 6 28 15 5 25
11 8 15 34 10 7 0 18 17 10 22 10 22 2 38 0 1 27 7 50 6 10 19 15 4 18 23 10 8 11 20 35
12 12 8 7 8 21 8 1 12 18 24 1 4 6 6 14 12 16 14 4 4 8 9 1 9 10 1 1 13 3 15 18
13 11 8 17 10 21 8 27 6 0 8 8 26 18 4 4 24 10 0 0 22 3 6 14 17 11 12 0 34 1 2 23
14 3 15 35 0 27 5 18 16 15 33 2 39 0 10 17 0 23 11 32 2 29 6 1 15 17 27 5 17 6 11 17
15 0 17 28 7 35 0 33 17 9 25 10 9 15 24 17 8 26 15 50 0 31 19 0 34 9 23 5 31 4 23 41
16 0 15 21 15 24 9 14 8 12 21 14 20 14 37 0 19 24 13 24 0 36 10 19 31 6 21 8 35 15 8 26
17 1 6 12 16 15 15 27 6 2 19 0 5 30 35 0 15 10 25 18 21 35 3 12 22 6 35 6 11 18 0 21
18 4 0 15 12 8 14 29 3 0 27 11 22 5 18 8 8 8 8 28 11 4 18 27 7 1 9 7 2 18 14 5

Arrival rates for large-scale experiments with 50 containers:

1 30 180 30 60 30 0 120 30 0 60 30 0 0 120 30 120 30 180 30 60
2 30 120 30 60 60 30 0 0 0 60 30 0 0 0 0 0 0 120 30 0
3 90 30 60 30 0 60 30 0 0 0 0 60 30 60 30 0 0 0 0 0
4 90 0 30 0 0 60 30 120 30 120 30 0 0 0 0 120 90 30 60 0
5 60 30 60 0 30 0 90 30 0 0 60 60 30 60 60 30 0 180 90 0
6 0 0 180 90 60 0 150 90 30 0 0 0 0 120 0 30 0 180 30 180
7 60 30 0 0 0 0 30 0 0 0 60 30 0 0 180 30 0 0 0 0
8 0 0 0 60 30 0 30 0 0 0 60 60 30 0 0 0 60 120 90 30
9 60 30 120 30 0 0 90 30 0 0 60 30 0 0 120 30 0 180 30 0
10 30 0 60 0 0 0 30 120 30 0 120 90 60 30 120 30 120 60 30 120
11 60 30 0 120 30 0 60 30 0 120 30 60 60 30 0 60 0 30 0 0
12 30 60 30 60 30 0 0 0 60 30 0 0 60 30 0 0 180 30 0 0
13 60 30 60 30 0 0 0 0 0 120 90 30 0 0 0 60 30 0 0 0
14 0 120 30 120 30 60 30 60 30 120 90 30 60 30 120 30 120 60 30 0
15 0 120 30 60 30 60 30 0 60 0 30 0 0 120 30 0 180 30 120 0
16 0 180 150 0 30 120 30 120 30 0 0 60 60 30 60 30 0 180 90 120
17 120 60 0 0 150 0 30 120 30 120 0 0 0 0 0 0 0 0 60 150 60
18 0 30 180 150 90 0 30 120 30 0 120 90 60 30 0 0 120 90 120 90
19 30 0 60 30 0 0 90 30 0 0 0 0 60 30 0 0 0 180 30 0
20 0 60 30 0 60 0 30 0 0 60 30 120 30 0 180 90 30 120 30 0
21 30 0 120 30 0 0 30 60 30 0 0 0 0 0 60 30 0 0 0 0
22 60 60 0 0 30 0 30 60 60 30 120 0 30 60 30 0 120 60 0 150
23 0 120 30 0 60 0 150 30 0 120 30 60 30 120 30 60 30 120 30 0
24 120 30 0 60 30 0 30 120 90 30 60 30 0 60 60 30 60 120 30 0
25 0 60 30 120 30 0 30 0 60 30 0 0 0 60 0 30 0 180 30 60
26 30 0 180 90 30 0 90 30 0 0 60 30 0 120 30 0 120 60 0 0
27 30 0 120 30 0 0 150 30 0 0 60 30 0 0 0 60 30 180 30 60
28 30 60 60 90 0 0 30 60 30 0 0 120 0 30 60 30 0 180 30 120
29 120 30 60 0 60 0 30 120 30 120 90 30 0 120 30 0 0 180 90 120
30 30 0 0 60 30 0 90 30 0 60 30 0 120 30 0 0 180 30 180 30
31 60 30 180 0 90 0 150 0 30 120 90 60 0 30 120 30 120 60 0 0
32 60 30 0 180 30 0 30 0 0 60 30 0 0 120 30 0 120 30 0 60
33 60 0 90 60 0 0 150 30 60 60 30 120 0 30 120 30 120 0 90 120
34 30 0 120 30 0 0 90 30 0 60 0 30 0 0 60 30 0 180 30 60
35 60 30 180 30 120 0 150 30 60 30 120 30 0 0 0 0 0 180 30 60
36 30 0 120 30 0 0 30 60 30 60 30 0 120 30 120 60 30 0 60 30
37 0 120 90 120 150 0 150 30 120 30 60 30 60 60 90 60 30 180 0 90
38 60 30 120 90 0 0 30 60 30 120 30 0 120 30 60 30 0 180 150 30
39 60 30 120 30 60 30 0 120 30 60 30 60 0 30 0 0 60 30 120 30
40 60 30 120 90 60 90 30 60 30 120 30 0 60 30 120 30 120 60 90 60
41 30 60 0 30 60 0 0 0 30 0 120 30 60 30 120 30 120 0 0 30
42 30 180 30 180 30 0 0 0 60 60 30 0 120 30 60 30 120 90 60 0
43 60 0 0 0 30 0 0 0 60 30 0 0 0 60 60 30 180 30 0 0
44 0 120 30 180 30 120 30 60 30 120 30 0 120 30 120 30 0 180 30 120
45 0 180 30 120 30 60 0 0 30 0 0 60 30 60 0 30 120 0 90 60
46 30 0 120 90 0 0 30 120 30 0 120 30 120 30 120 30 0 180 150 90
47 0 30 60 120 30 0 30 0 0 120 30 0 120 0 30 120 0 30 120 30
48 30 60 30 60 30 0 150 90 30 120 30 0 0 60 30 120 30 0 0 60

49 60 30 120 90 30 0 30 120 30 120 30 120 30 120 30 120 30 180 150 90
50 30 0 120 30 0 0 150 0 30 0 60 30 0 120 30 0 120 30 120 90

Arrival rates for large-scale experiments with 50 containers and increased arrival rates by 10%:

1 33 198 33 66 33 0 132 33 0 66 33 0 0 132 33 132 33 198 33 66
2 33 132 33 66 66 33 0 0 0 66 33 0 0 0 0 0 0 132 33 0
3 99 33 66 33 0 66 33 0 0 0 0 66 33 66 33 0 0 0 0 0
4 99 0 33 0 0 66 33 132 33 132 33 0 0 0 0 132 99 33 66 0
5 66 33 66 0 33 0 99 33 0 0 66 66 33 66 66 33 0 198 99 0
6 0 0 198 99 66 0 165 99 33 0 0 0 0 132 0 33 0 198 33 198
7 66 33 0 0 0 0 33 0 0 0 66 33 0 0 198 33 0 0 0 0
8 0 0 0 66 33 0 33 0 0 0 66 66 33 0 0 0 66 132 99 33
9 66 33 132 33 0 0 99 33 0 0 66 33 0 0 132 33 0 198 33 0
10 33 0 66 0 0 0 33 132 33 0 132 99 66 33 132 33 132 66 33 132
11 66 33 0 132 33 0 66 33 0 132 33 66 66 33 0 66 0 33 0 0
12 33 66 33 66 33 0 0 0 66 33 0 0 66 33 0 0 198 33 0 0
13 66 33 66 33 0 0 0 0 0 132 99 33 0 0 0 66 33 0 0 0
14 0 132 33 132 33 66 33 66 33 132 99 33 66 33 132 33 132 66 33 0
15 0 132 33 66 33 66 33 0 66 0 33 0 0 132 33 0 198 33 132 0
16 0 198 165 0 33 132 33 132 33 0 0 66 66 33 66 33 0 198 99 132
17 132 66 0 0 165 0 33 132 33 132 0 0 0 0 0 0 0 0 66 165 66
18 0 33 198 165 99 0 33 132 33 0 132 99 66 33 0 0 132 99 132 99
19 33 0 66 33 0 0 99 33 0 0 0 0 66 33 0 0 0 198 33 0
20 0 66 33 0 66 0 33 0 0 66 33 132 33 0 198 99 33 132 33 0
21 33 0 132 33 0 0 33 66 33 0 0 0 0 0 66 33 0 0 0 0
22 66 66 0 0 33 0 33 66 66 33 132 0 33 66 33 0 132 66 0 165
23 0 132 33 0 66 0 165 33 0 132 33 66 33 132 33 66 33 132 33 0
24 132 33 0 66 33 0 33 132 99 33 66 33 0 66 66 33 66 132 33 0
25 0 66 33 132 33 0 33 0 66 33 0 0 0 66 0 33 0 198 33 66
26 33 0 198 99 33 0 99 33 0 0 66 33 0 132 33 0 132 66 0 0
27 33 0 132 33 0 0 165 33 0 0 66 33 0 0 0 66 33 198 33 66
28 33 66 66 99 0 0 33 66 33 0 0 132 0 33 66 33 0 198 33 132
29 132 33 66 0 66 0 33 132 33 132 99 33 0 132 33 0 0 198 99 132
30 33 0 0 66 33 0 99 33 0 66 33 0 132 33 0 0 198 33 198 33
31 66 33 198 0 99 0 165 0 33 132 99 66 0 33 132 33 132 66 0 0
32 66 33 0 198 33 0 33 0 0 66 33 0 0 132 33 0 132 33 0 66
33 66 0 99 66 0 0 165 33 66 66 33 132 0 33 132 33 132 0 99 132
34 33 0 132 33 0 0 99 33 0 66 0 33 0 0 66 33 0 198 33 66
35 66 33 198 33 132 0 165 33 66 33 132 33 0 0 0 0 0 198 33 66
36 33 0 132 33 0 0 33 66 33 66 33 0 132 33 132 66 33 0 66 33
37 0 132 99 132 165 0 165 33 132 33 66 33 66 66 99 66 33 198 0 99
38 66 33 132 99 0 0 33 66 33 132 33 0 132 33 66 33 0 198 165 33
39 66 33 132 33 66 33 0 132 33 66 33 66 0 33 0 0 66 33 132 33
40 66 33 132 99 66 99 33 66 33 132 33 0 66 33 132 33 132 66 99 66
41 33 66 0 33 66 0 0 0 33 0 132 33 66 33 132 33 132 0 0 33
42 33 198 33 198 33 0 0 0 66 66 33 0 132 33 66 33 132 99 66 0
43 66 0 0 0 33 0 0 0 66 33 0 0 0 66 66 33 198 33 0 0
44 0 132 33 198 33 132 33 66 33 132 33 0 132 33 132 33 0 198 33 132
45 0 198 33 132 33 66 0 0 33 0 0 66 33 66 0 33 132 0 99 66

46 33 0 132 99 0 0 33 132 33 0 132 33 132 33 132 33 0 198 165 99
47 0 33 66 132 33 0 33 0 0 132 33 0 132 0 33 132 0 33 132 33
48 33 66 33 66 33 0 165 99 33 132 33 0 0 66 33 132 33 0 0 66
49 66 33 132 99 33 0 33 132 33 132 33 132 33 132 33 132 33 198 165 99
50 33 0 132 33 0 0 165 0 33 0 66 33 0 132 33 0 132 33 132 99

Arrival rates for large-scale experiments with 50 containers and decreased arrival rates by 25%:

1 22 135 22 45 22 0 90 22 0 45 22 0 0 90 22 90 22 135 22 45
2 22 90 22 45 45 22 0 0 0 45 22 0 0 0 0 0 0 90 22 0
3 67 22 45 22 0 45 22 0 0 0 0 45 22 45 22 0 0 0 0 0
4 67 0 22 0 0 45 22 90 22 90 22 0 0 0 0 90 67 22 45 0
5 45 22 45 0 22 0 67 22 0 0 45 45 22 45 45 22 0 135 67 0
6 0 0 135 67 45 0 112 67 22 0 0 0 0 90 0 22 0 135 22 135
7 45 22 0 0 0 0 22 0 0 0 45 22 0 0 135 22 0 0 0 0
8 0 0 0 45 22 0 22 0 0 0 45 45 22 0 0 0 45 90 67 22
9 45 22 90 22 0 0 67 22 0 0 45 22 0 0 90 22 0 135 22 0
10 22 0 45 0 0 0 22 90 22 0 90 67 45 22 90 22 90 45 22 90
11 45 22 0 90 22 0 45 22 0 90 22 45 45 22 0 45 0 22 0 0
12 22 45 22 45 22 0 0 0 45 22 0 0 45 22 0 0 135 22 0 0
13 45 22 45 22 0 0 0 0 0 90 67 22 0 0 0 45 22 0 0 0
14 0 90 22 90 22 45 22 45 22 90 67 22 45 22 90 22 90 45 22 0
15 0 90 22 45 22 45 22 0 45 0 22 0 0 90 22 0 135 22 90 0
16 0 135 112 0 22 90 22 90 22 0 0 45 45 22 45 22 0 135 67 90
17 90 45 0 0 112 0 22 90 22 90 0 0 0 0 0 0 0 0 45 112 45
18 0 22 135 112 67 0 22 90 22 0 90 67 45 22 0 0 90 67 90 67
19 22 0 45 22 0 0 67 22 0 0 0 0 45 22 0 0 0 135 22 0
20 0 45 22 0 45 0 22 0 0 45 22 90 22 0 135 67 22 90 22 0
21 22 0 90 22 0 0 22 45 22 0 0 0 0 0 45 22 0 0 0 0
22 45 45 0 0 22 0 22 45 45 22 90 0 22 45 22 0 90 45 0 112
23 0 90 22 0 45 0 112 22 0 90 22 45 22 90 22 45 22 90 22 0
24 90 22 0 45 22 0 22 90 67 22 45 22 0 45 45 22 45 90 22 0
25 0 45 22 90 22 0 22 0 45 22 0 0 0 45 0 22 0 135 22 45
26 22 0 135 67 22 0 67 22 0 0 45 22 0 90 22 0 90 45 0 0
27 22 0 90 22 0 0 112 22 0 0 45 22 0 0 0 45 22 135 22 45
28 22 45 45 67 0 0 22 45 22 0 0 90 0 22 45 22 0 135 22 90
29 90 22 45 0 45 0 22 90 22 90 67 22 0 90 22 0 0 135 67 90
30 22 0 0 45 22 0 67 22 0 45 22 0 90 22 0 0 135 22 135 22
31 45 22 135 0 67 0 112 0 22 90 67 45 0 22 90 22 90 45 0 0
32 45 22 0 135 22 0 22 0 0 45 22 0 0 90 22 0 90 22 0 45
33 45 0 67 45 0 0 112 22 45 45 22 90 0 22 90 22 90 0 67 90
34 22 0 90 22 0 0 67 22 0 45 0 22 0 0 45 22 0 135 22 45
35 45 22 135 22 90 0 112 22 45 22 90 22 0 0 0 0 0 135 22 45
36 22 0 90 22 0 0 22 45 22 45 22 0 90 22 90 45 22 0 45 22
37 0 90 67 90 112 0 112 22 90 22 45 22 45 45 67 45 22 135 0 67
38 45 22 90 67 0 0 22 45 22 90 22 0 90 22 45 22 0 135 112 22
39 45 22 90 22 45 22 0 90 22 45 22 45 0 22 0 0 45 22 90 22
40 45 22 90 67 45 67 22 45 22 90 22 0 45 22 90 22 90 45 67 45
41 22 45 0 22 45 0 0 0 22 0 90 22 45 22 90 22 90 0 0 22
42 22 135 22 135 22 0 0 0 45 45 22 0 90 22 45 22 90 67 45 0

43 45 0 0 0 22 0 0 0 45 22 0 0 0 45 45 22 135 22 0 0

44 0 90 22 135 22 90 22 45 22 90 22 0 90 22 90 22 0 135 22 90

45 0 135 22 90 22 45 0 0 22 0 0 45 22 45 0 22 90 0 67 45

46 22 0 90 67 0 0 22 90 22 0 90 22 90 22 90 22 0 135 112 67

47 0 22 45 90 22 0 22 0 0 90 22 0 90 0 22 90 0 22 90 22

48 22 45 22 45 22 0 112 67 22 90 22 0 0 45 22 90 22 0 0 45

49 45 22 90 67 22 0 22 90 22 90 22 90 22 90 22 90 22 135 112 67

50 22 0 90 22 0 0 112 0 22 0 45 22 0 90 22 0 90 22 90 67

Mathematical Optimization Model

To solve the mathematical model exactly, it was implemented using python:

```
!pip install pulp
import pulp
import math


def compute_predicted_fills_and_must_visit(current_fill_kg, arrival_rate_kg, bin_capacity_kg, threshold=0.85):

    n = len(current_fill_kg)
    predicted_fill = [0.0] * (n + 1)
    must_visit = [0] * (n + 1)
    for i in range(1, n+1):
        cur = current_fill_kg[i-1]
        arr = arrival_rate_kg[i-1]
        cap = bin_capacity_kg[i-1]
        pred = cur + arr
        pred = max(0.0, pred)
        predicted_fill[i] = pred
        must_visit[i] = 1 if pred >= threshold * cap else 0
    must_visit[0] = 0
    predicted_fill[0] = 0.0
    return predicted_fill, must_visit

def solve_cvrp_pulp_with_mustvisit(distance_matrix, predicted_fill, must_visit, K, Q, cost_per_km=1.0, time_limit=None):

    n_plus_1 = len(distance_matrix)
    n = n_plus_1 - 1

    total_must_demand = sum(predicted_fill[i] for i in range(1, n_plus_1) if must_visit[i] == 1)
    if total_must_demand > K * Q + 1e-9:
        raise ValueError(f"Infeasible: total must-visit demand {total_must_demand:.2f} kg > fleet capacity {K*Q} kg")

    prob = pulp.LpProblem("CVRP_mustvisit", pulp.LpMinimize)

    x = {}
    for i in range(n_plus_1):
        for j in range(n_plus_1):
            if i != j:
                x[(i, j)] = pulp.LpVariable(f"x_{i}_{j}", cat="Binary")

    u = {}
    for i in range(1, n_plus_1):
        lb = predicted_fill[i] * must_visit[i]
        u[i] = pulp.LpVariable(f"u_{i}", lowBound=0.0, upBound=Q, cat="Continuous")

    prob += pulp.lpSum(distance_matrix[i][j] * x[(i, j)] for i in range(n_plus_1) for j in range(n_plus_1) if i != j) * cost_per_km

    for i in range(1, n_plus_1):
        prob += pulp.lpSum(x[(i, j)] for j in range(n_plus_1) if j != i) == must_visit[i], f"out_deg_{i}"
        prob += pulp.lpSum(x[(j, i)] for j in range(n_plus_1) if j != i) == must_visit[i], f"in_deg_{i}"

    prob += pulp.lpSum(x[(0, j)] for j in range(1, n_plus_1)) <= K, "depot_out"
    prob += pulp.lpSum(x[(i, 0)] for i in range(1, n_plus_1)) <= K, "depot_in"
```

```
for i in range(1, n_plus_1):
    for j in range(1, n_plus_1):
        if i != j:
            prob += u[i] - u[j] + Q * x[(i, j)] <= Q - predicted_fill[j], f"mtz_{i}_{j}"

for i in range(1, n_plus_1):
    prob += u[i] <= Q * must_visit[i], f"u_zero_if_notvisited_{i}"

solver = pulp.PULP_CBC_CMD(msg=True, timeLimit=time_limit) if time_limit else pulp.PULP_CBC_CMD(msg=True)
result_status = prob.solve(solver)

status = pulp.LpStatus[prob.status]
print("Solver status:", status)

if status not in ("Optimal", "Feasible"):
    print("No feasible/optimal solution found.")
    return None

succ = {i: [] for i in range(n_plus_1)}
for i in range(n_plus_1):
    for j in range(n_plus_1):
        if i != j and pulp.value(x[(i, j)]) is not None and pulp.value(x[(i, j)]) > 0.5:
            succ[i].append(j)

routes = []
visited_nodes = set()
for first in succ[0]:
    route = [0, first]
    visited_nodes.add(first)
    cur = first
    while True:
        next_nodes = [j for j in succ[cur] if j != 0]
        if not next_nodes:
            if 0 in succ[cur]:
                route.append(0)
            break
        nxt = next_nodes[0]
        route.append(nxt)
        visited_nodes.add(nxt)
        cur = nxt
        if cur == 0:
            break
    routes.append(route)

total_distance = sum(distance_matrix[i][j] * pulp.value(x[(i, j)]) for i in range(n_plus_1) for j in range(n_plus_1) if i != j)
total_cost = total_distance * cost_per_km

print(f"Total distance: {total_distance:.2f}, Total cost: {total_cost:.2f}")
print(f"Number of routes used: {len(routes)}")
print("Routes (node indices):")
for r in routes:
    print(r)

   return {
       "status": status,
       "routes": routes,
       "total_distance": total_distance,
       "total_cost": total_cost,
       "predicted_fill": predicted_fill,
       "must_visit": must_visit
   }


if __name__ == "__main__":

    distance_matrix = []
    current_fill_kg = []
    arrival_rate_kg = []
    bin_capacity_kg = []

    K = 2
    Q = 4800
    cost_per_km = 1.0

    predicted_fill, must_visit = compute_predicted_fills_and_must_visit(current_fill_kg, arrival_rate_kg, bin_capacity_kg, threshold=0.85)
    print("Predicted fills (kg):", predicted_fill)
    print("Must visit containers:", must_visit)

    result = solve_cvrp_pulp_with_mustvisit(distance_matrix, predicted_fill, must_visit, K, Q, cost_per_km=1.0, time_limit=60)
```

Simulation

The simulation part contains several classes, to anticipate future container fill levels by considering historical data and to determine optimal routes based on the predicted fill levels aiming to minimize total travel distance while avoiding container overflows.

First, global variables are needed ensuring that constraints can easily be adapted during testing procedures:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Waste_Collection
{
    internal class GlobalVariables
    {
        public const int PLANNING_HORIZON = 15;
        public const int MAX_VEHICLE_CAPACITY = 1623;
        public const int BIN_LEVEL_THRESHOLD = 85;
    }
}

```

The class Filehandler enables reading the datasets and transforms data into objects:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Waste_Collection
{
    internal class Filehandler
    {

        public Filehandler()
        {

        }
        public int[,] Read(string path)
        {
            string[] input = File.ReadAllLines(path);
            int x = input.Length;
            int y = 0;
            foreach (string line in input)
            {
                string[] splitted = line.Split(' ');
                y = splitted.Length;
            }

            int[,] arrivalRates = new int[x,y];
            int index = 0;


            foreach (string s in input)
            {
                string[] splitted = s.Split(' ');

                for (int i = 0; i < splitted.Length; i++)
                {
                    arrivalRates[index, i] = int.Parse(splitted[i]);
                }
                index++;
            }
            return arrivalRates;
        }
```

```
        public double[,] ReadDistanceMatrix(string path)
        {
            string[] input = File.ReadAllLines(path);
            int x = input.Length;
            int y = 0;
            foreach (string line in input)
            {
                string[] splitted = line.Split(',');
                y = splitted.Length;
            }

            double[,] arrivalRates = new double[x, y];
            int index = 0;

            foreach (string s in input)
            {
                string[] splitted = s.Split(',');

                for (int i = 0; i < splitted.Length; i++)
                {
                    arrivalRates[index, i] = double.Parse(splitted[i],
                    System.Globalization.CultureInfo.InvariantCulture);
                }
                index++;
            }
            return arrivalRates;
        }
    }
}
```

To ensure that the container fill levels can be monitored during simulation, a container class is implemented. Additionally, the waste container objects maintain the average arrival rate and standard deviation, which are required for the predictive model to determine forecasts:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using static System.Runtime.InteropServices.JavaScript.JSType;

namespace Waste_Collection
{
    internal class WasteContainer
    {
        public int Id { get; set; }
        public int FillLevel {  get; set; }
        public List<int> ArrivalRates { get; set; }
        public int AvgArrivalRate {  get; set; }
        public int StdDev {  get; set; }
        public WasteContainer(int id) {
            Id = id;
            FillLevel = 0;
            AvgArrivalRate = 0;
            StdDev = 0;
        }
        public WasteContainer(int id, int fill_level)
        {
            Id = id;
            FillLevel = fill_level;
        }
        public WasteContainer(int id, int fill_level, int arrivalRate)
        {
            Id = id;
            FillLevel = fill_level;
            AvgArrivalRate = arrivalRate;
            ArrivalRates = new List<int>();
            ArrivalRates.Add(arrivalRate);
            StdDev = 0;
        }
        public WasteContainer(int id, int fill_level, int arrivalRate, List<int>
        arrivalRates)
        {
            Id = id;
            FillLevel = fill_level;
            AvgArrivalRate = arrivalRate;
            ArrivalRates = new List<int>();
            ArrivalRates.AddRange(arrivalRates);
            ArrivalRates.Add(arrivalRate);
            StdDev = 0;
        }
```

```
        public void AddWaste(int newWaste)
        {
            FillLevel += newWaste;
            ArrivalRates.Add((int)newWaste);

            double mean = 0;
            foreach (var d in ArrivalRates)
                mean += d;
            mean /= ArrivalRates.Count;
            AvgArrivalRate = Convert.ToInt32(mean);

            double variance = 0;
            foreach (var d in ArrivalRates)
                variance += Math.Pow(d - mean, 2);
            double stdDev = Math.Sqrt(variance / ArrivalRates.Count);
            StdDev = Convert.ToInt32((double)stdDev);
        }
        public void RemoveWaste()
        {
            FillLevel = 0;
        }
        public override string ToString()
        {
            return $"{Id} - Fill level: {FillLevel} - avg. arrival rate:
            {AvgArrivalRate}";
        }

    }
}

```

The class Route ensures that a planned route consisting of several waste container objects can easily be adapted by rearranging containers aiming to decrease total travel distance. Furthermore, the route also maintains important characteristics like total route length and the expected amount of waste that will be collected by a vehicle.

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Waste_Collection
{
    internal class Route
    {
        public int ID { get; set; }
        public List<WasteContainer> ContainerTour;
        public List<WasteContainer> UnvisitedContainers;
        public double[,] DistanceMatrix;
        public double TotalTourCosts {  get; set; }
        public Route(int id, double[,] distanceMatrix)
        {
            ID = id;
            ContainerTour = new List<WasteContainer>();
            UnvisitedContainers = new List<WasteContainer>();
            DistanceMatrix = distanceMatrix;
            TotalTourCosts = 0;
        }
        public Route(List<WasteContainer> containerTour, double[,] distanceMatrix)
        {
            ContainerTour = containerTour;
            UnvisitedContainers = new List<WasteContainer>();
            DistanceMatrix = distanceMatrix;
            TotalTourCosts = 0;
            CalculateTourCosts();
        }
        public Route(int id, List<WasteContainer> containerTour, double[,]
        distanceMatrix)
        {
            ID = id;
            ContainerTour = containerTour;
            UnvisitedContainers = new List<WasteContainer>();
            DistanceMatrix = distanceMatrix;
            TotalTourCosts = 0;
            CalculateTourCosts();
        }
        public void AddContainer(WasteContainer c)
        {
            ContainerTour.Add(c);
            CalculateTourCosts();
        }
```

```
        public void CalculateTourCosts()
        {
            TotalTourCosts = 0;
            if(ContainerTour.Count > 0)
            {
                TotalTourCosts += DistanceMatrix[0, ContainerTour[0].Id];
                for (int i = 0; i < ContainerTour.Count - 1; i++)
                {
                    TotalTourCosts += DistanceMatrix[ContainerTour[i].Id,
                    ContainerTour[i + 1].Id];
                }
                TotalTourCosts += DistanceMatrix[ContainerTour[ContainerTour.Count -
                1].Id, 0];
            }

        }
        public int GetTourFillLevel()
        {
            int fillLevel = 0;
            foreach(WasteContainer c in ContainerTour)
            {
                fillLevel += c.FillLevel;
            }
            return fillLevel;
        }
        public override string ToString()

        {
            StringBuilder stringBuilder = new StringBuilder();
            stringBuilder.Append("ID: " + ID + " - Containers: ");
            foreach (WasteContainer p in ContainerTour)
            {
                stringBuilder.Append(p.Id + " ");
            }
            stringBuilder.Append("\t => distance: " + Math.Round(TotalTourCosts, 2));
            stringBuilder.Append("\t => total fill level: " + GetTourFillLevel());
            return stringBuilder.ToString();
        }
    }
}
```

The class Forecasting anticipates container fill levels based on historical data and results in a forecasting matrix that will be used by the routing algorithm in the next step.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Waste_Collection
{
    internal class Forecasting
    {
        public Forecasting() { }

        public int[,] Forecast(List<WasteContainer> containerList, int[,]
        arrivalRates, int periods)
        {
            int[,] forecastingList = new int[containerList.Count, periods];
            List<WasteContainer> tmpContainerList = containerList;

            for (int i = 0; i < forecastingList.GetLength(0); i++)
            {
                for (int j = 0; j < forecastingList.GetLength(1); j++)
                {
                    if (j == 0)
                    {
                        tmpContainerList[i].AddWaste(arrivalRates[i, 1]);
                        forecastingList[i, j] = tmpContainerList[i].FillLevel;
                    }
                    else
                    {

                        tmpContainerList[i].AddWaste(tmpContainerList[i].AvgArrivalRat
                        e);
                        forecastingList[i,j] += tmpContainerList[i].FillLevel;
                    }
                }
            }

            return forecastingList;
        }
```

```
37          public int[,] StochasticForecast(List<WasteContainer> containerList,
            List<int> arrivalRates)
38          {
39              int[,] forecastingList = new int[containerList.Count,
                GlobalVariables.PLANNING_HORIZON];
40              List<WasteContainer> tmpContainerList = new List<WasteContainer>();
41
42              foreach (WasteContainer container in containerList)
43              {
44                  WasteContainer w = new WasteContainer(container.Id,
                    container.FillLevel, container.AvgArrivalRate,
                    container.ArrivalRates);
45                  tmpContainerList.Add(w);
46              }
47
48              Random rand = new Random();
49
50              for (int i = 0; i < forecastingList.GetLength(0); i++)
51              {
52                  for (int j = 0; j < forecastingList.GetLength(1); j++)
53                  {
54                      if (j == 0)
55                      {
56                          forecastingList[i, j] = tmpContainerList[i].FillLevel;
57                      }
58                      else
59                      {
60                          double u1 = 1.0 - rand.NextDouble();
61                          double u2 = 1.0 - rand.NextDouble();
62                          double randStdNormal = Math.Sqrt(-2.0 * Math.Log(u1)) *
63                                                 Math.Sin(2.0 * Math.PI * u2);
64
65                          double forecast = tmpContainerList[i].AvgArrivalRate +
                            tmpContainerList[i].StdDev * randStdNormal;
66
                             tmpContainerList[i].AddWaste(Convert.ToInt32(Math.Max(forecast
                            , 0)));
67                          forecastingList[i, j] += tmpContainerList[i].FillLevel;
68                      }
69                  }
70              }
71
72              return forecastingList;
73          }
74
75          public SortedList<int, List<int>> DetermineMustGoContainers(int[,]
            forecastingList)
76          {
77              int[,] mustGoSchedule = new int[7,5];
78              int threshold = 60;
79              SortedList<int, List<int>> sortedList = new SortedList<int, List<int>>();
80              List<int> alreadyPlanned = new List<int>();
81
82              for(int i = 0; i <7; i++)
83              {
84                  sortedList.Add(i, new List<int>());
85              }
86
87
88              for (int i = 0; i < forecastingList.GetLength(0); i++)
89              {
90                  for (int j = 0; j < forecastingList.GetLength(1); j++)
91                  {
92                      if ((forecastingList[i,j] >= threshold) &&
                        !alreadyPlanned.Contains(i))
93                      {
94                          sortedList.ElementAt(j).Value.Add(i);
95                          alreadyPlanned.Add(i);
96                      }
97
98                  }
99              }
100
101             return sortedList;
102         }
103
104         public List<Route> DetermineMustGos(List<WasteContainer> containerList,
            int[,] forecastingList, double[,] distanceMatrix)
105         {
106             int[,] mustGoSchedule = new int[GlobalVariables.PLANNING_HORIZON, 5];
107             int threshold = GlobalVariables.BIN_LEVEL_THRESHOLD;
108             List<Route> routes = new List<Route>();
109             List<int> alreadyPlanned = new List<int>();
110
111             for (int i = 0; i < GlobalVariables.PLANNING_HORIZON; i++)
112             {
113                 routes.Add(new Route(i, distanceMatrix));
114             }
115
116             for (int i = 0; i < forecastingList.GetLength(0); i++)
117             {
118                 for (int j = 0; j < forecastingList.GetLength(1); j++)
119                 {
120                     if ((forecastingList[i, j] >= threshold) &&
                        !alreadyPlanned.Contains(i))
121                     {
122
123                         routes.ElementAt(j).AddContainer(containerList[i]);
124                         alreadyPlanned.Add(i);
125                     }
126
127                 }
128             }
129
130             return routes;
131         }
132
```

```
    }
}

```

The class routing entails the algorithm to determine most optimal routes by avoiding overflowing containers based on the forecasts determined by the predictive model.

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Waste_Collection
{
    internal class Routing
    {
        public double[,] DistanceMatrix { get; set; }
        public Routing(double[,] distanceMatrix)
        {
            DistanceMatrix = distanceMatrix;
        }
        public Route NearestNeighbor(int id, List<WasteContainer> visitContainers)
        {
            Route tour = new Route(id, DistanceMatrix);

            WasteContainer depot = new WasteContainer(0,0);
            double costs = double.MaxValue;
            double currCosts =  0;

            foreach (WasteContainer c in visitContainers)
                tour.UnvisitedContainers.Add(c);

            //Create tour with depot as starting point
            tour.ContainerTour.Add(depot);

            // All not visited points are stored in the initial points
            // We continue, while we have unvisited points
            while(tour.UnvisitedContainers.Count > 0)
            {
                WasteContainer cheapestContainer = new WasteContainer(50);
                for (int i = 0; i < tour.UnvisitedContainers.Count; i++)
                {
                    currCosts =
                    DistanceMatrix[tour.ContainerTour[tour.ContainerTour.Count -
                    1].Id, tour.UnvisitedContainers[i].Id];
                    if (currCosts < costs)
                    {
                        cheapestContainer = tour.UnvisitedContainers[i];
                        costs = currCosts;
                    }
                }
                tour.ContainerTour.Add(cheapestContainer);
                tour.UnvisitedContainers.Remove(cheapestContainer);
                costs = double.MaxValue;
            }
            tour.ContainerTour.Add(depot);
            tour.CalculateTourCosts();
            return tour;
        }
        public List<Route> CheckRoutesForDepot(List<Route> nearestNeighborRoutes)
        {
            List<Route> adaptedRoutes = nearestNeighborRoutes;

            foreach (Route route in adaptedRoutes)
            {
                Console.WriteLine("Route Tour Fill Level: "+route.GetTourFillLevel());
                if (route.GetTourFillLevel() > GlobalVariables.MAX_VEHICLE_CAPACITY)
                {
                    int tmpFill = 0;
                    Console.WriteLine("tmpFill Level: " + tmpFill);
                    for (int i = 0; i < route.ContainerTour.Count; i++)
                    {
                        tmpFill += route.ContainerTour[i].FillLevel;

                        if (tmpFill > GlobalVariables.MAX_VEHICLE_CAPACITY)
                        {
                            route.ContainerTour.Insert(i, new WasteContainer(0, 0));
                            tmpFill = route.ContainerTour[i].FillLevel;

                        }
                    }
                }
                route.CalculateTourCosts();
            }

            return adaptedRoutes;
        }


        public List<Route> VariableNeighborhoodSearch(List<Route>
        nearestNeighborRoutes)
        {
            List<Route> neighborhoodRoutes =
            CreateNewRouteList(nearestNeighborRoutes);
            List<Route> cheapestRoute = new List<Route>();
            double cheapestTourCosts = double.MaxValue;
            double previousTourCosts = CalculateCostsOfTour(neighborhoodRoutes);
            double currTourCosts = double.MaxValue;
```

```
89
90              for(int x = 0; x <
                DetermineNeighborhoodSearchIterations(neighborhoodRoutes)*10; x++)
91              {
92                  List<Route> tmp = CreateNewRouteList(neighborhoodRoutes);
93                  int cheapestRouteIndex = SmallestRoute(neighborhoodRoutes);
94                  currTourCosts = 0;
95                  int index = cheapestRouteIndex - 1;
96
97
98                  for (int i = 1; i <
                    nearestNeighborRoutes[cheapestRouteIndex].ContainerTour.Count - 1;
                    i++)
99                  {
100                     int moveToIndex = cheapestRouteIndex;
101
102                     for (int j = cheapestRouteIndex - 1; j >= 0; j--)
103                     {
104                         if (neighborhoodRoutes[j].GetTourFillLevel() > 0 &&
                            neighborhoodRoutes[j].GetTourFillLevel() +
                            nearestNeighborRoutes[cheapestRouteIndex].ContainerTour[i].Fil
                            lLevel <= GlobalVariables.MAX_VEHICLE_CAPACITY)
105                         {
106                             moveToIndex = j;
107                             break;
108                         }
109                     }
110                     if (moveToIndex != cheapestRouteIndex &&
                        !neighborhoodRoutes[moveToIndex].ContainerTour.Contains(nearestNei
                        ghborRoutes[cheapestRouteIndex].ContainerTour[i]))
111                     {
112                         int insertionIndex =
                            CheapestInsertion(neighborhoodRoutes[moveToIndex],
                            nearestNeighborRoutes[cheapestRouteIndex].ContainerTour[i]);
113
                            neighborhoodRoutes[moveToIndex].ContainerTour.Insert(insertion
                            Index,
                            nearestNeighborRoutes[cheapestRouteIndex].ContainerTour[i]);
114
                            neighborhoodRoutes[cheapestRouteIndex].ContainerTour.Remove(ne
                            arestNeighborRoutes[cheapestRouteIndex].ContainerTour[i]);
115
116
117                     }
118                 }
119                 currTourCosts = CalculateCostsOfTour(neighborhoodRoutes);
120                 if (currTourCosts < cheapestTourCosts)
121                 {
122                     cheapestRoute = neighborhoodRoutes;
123                     cheapestTourCosts = currTourCosts;
124                 }
125             }
126
127             return cheapestRoute;
128         }
129
130         public List<Route> Savingsalgorithm(List<Route> nearestNeighborRoutes)
131         {
132             List<Route> neighborhoodRoutes =
                CreateNewRouteList(nearestNeighborRoutes);
133             List<Route> tmp = CreateNewRouteList(nearestNeighborRoutes);
134             List<Route> savingsAlgorithmRoutes = new List<Route>();
135             bool exit = false;
136
137             int smallestRouteIndex = SmallestRoute(neighborhoodRoutes);
138             List<int> possibleInsertions = new List<int>();
139
140             foreach (Route r in nearestNeighborRoutes)
141             {
142                 if (r.GetTourFillLevel() > 0 && r.ID < smallestRouteIndex)
143                 {
144                     possibleInsertions.Add(r.ID);
145                 }
146             }
147
148             do
149             {
150
151             double bestSavings = 0;
152             int bestSavingsRouteIndex = 0;
153             int bestSavingsInsertionIndex = 0;
154
155                 if (possibleInsertions.Count > 0)
156                 {
157                     for (int i = 1; i < tmp[smallestRouteIndex].ContainerTour.Count -
                        1; i++)
158                     {
159                         for (int j = 0; j < possibleInsertions.Count; j++)
160                         {
161                             double currSavings =
                                DetermineSavingsOfRemoval(neighborhoodRoutes[smallestRoute
                                Index], tmp[smallestRouteIndex].ContainerTour[i]);
162                             double costsOfInsertion =
                                CostsOfInsertion(neighborhoodRoutes[possibleInsertions[j]]
                                , tmp[smallestRouteIndex].ContainerTour[i]);
163                             int cheapestInsertion =
                                CheapestInsertion(neighborhoodRoutes[possibleInsertions[j]
                                ], tmp[smallestRouteIndex].ContainerTour[i]);
164                             double currTotalSavings = currSavings - costsOfInsertion;
165                             //Console.WriteLine($"Savings of moving
                                {neighborhoodRoutes[smallestRouteIndex].ContainerTour[i]}
                                to route {possibleInsertions[j]} = {currTotalSavings}");
166                             if (currTotalSavings > 0 && currTotalSavings >
                                bestSavings && neighborhoodRoutes[j].GetTourFillLevel() +
                                tmp[smallestRouteIndex].ContainerTour[i].FillLevel <=
                                GlobalVariables.MAX_VEHICLE_CAPACITY)
```

```
                            {
                                bestSavings = currTotalSavings;
                                bestSavingsRouteIndex = possibleInsertions[j];
                                bestSavingsInsertionIndex = cheapestInsertion;
                            }
                            else
                            {
                                exit = true;
                            }


                        }
                        if (bestSavings > 0 && bestSavings < double.MaxValue)
                        {

                            neighborhoodRoutes[bestSavingsRouteIndex].ContainerTour.In
                            sert(bestSavingsInsertionIndex,
                            tmp[smallestRouteIndex].ContainerTour[i]);

                            neighborhoodRoutes[smallestRouteIndex].ContainerTour.Remov
                            e(tmp[smallestRouteIndex].ContainerTour[i]);

                        }
                    }

                    Console.WriteLine("After SavingsAlgorithm: ");
                    foreach(var route in neighborhoodRoutes)
                    {
                        Console.WriteLine(route);
                    }

                    tmp = CreateNewRouteList(neighborhoodRoutes);
                    smallestRouteIndex = SmallestRoute(tmp);
                    possibleInsertions.Clear();
                    foreach (Route r in tmp)
                    {
                        if (r.GetTourFillLevel() > 0 && r.ID < smallestRouteIndex)
                        {
                            possibleInsertions.Add(r.ID);
                        }
                    }
                }
            } while (possibleInsertions.Count > 0 && !exit);
            //test start
            Route test = NearestNeighbor(tmp.ElementAt(0).ID,
            tmp.ElementAt(0).ContainerTour);
            test.CalculateTourCosts();
            if(test.TotalTourCosts < tmp.ElementAt(0).TotalTourCosts)
            {
                tmp[0] = test;
            }
            //test ende
            return tmp;
        }
        public List<Route> DetermineCheapestAlgorithm(List<Route>
        nearestNeighborRoutes)
        {
            List<Route> cheapestRoute = new List<Route>();
            List<Route> variableNeighborhoodSearch =
            VariableNeighborhoodSearch(nearestNeighborRoutes);
            List<Route> savingsAlgorithm =
            Savingsalgorithm(variableNeighborhoodSearch);
            double tourCostsVNS = 0;
            double tourCostsSA = 0;

            foreach (var route in variableNeighborhoodSearch)
            {
                route.CalculateTourCosts();
                tourCostsVNS += route.TotalTourCosts;
            }
            foreach (var route in savingsAlgorithm)
            {
                route.CalculateTourCosts();
                tourCostsSA += route.TotalTourCosts;
            }
            if(tourCostsVNS < tourCostsSA)
            {
                Console.WriteLine($"Take VNS Tour \t Tour costs of VNS {tourCostsVNS}
                \t Tour costs of Savings Algorithm {tourCostsSA}");
                return variableNeighborhoodSearch;
            }
            else
            {
                Console.WriteLine($"Take Savings Algorithm Tour \t Tour costs of VNS
                {tourCostsVNS} \t Tour costs of Savings Algorithm {tourCostsSA}");
                return savingsAlgorithm;
            }

        }
        public double DetermineSavingsOfRemoval(Route route, WasteContainer
        wastecontainer)
        {
            double savingsOfRemoval = 0;

            for(int i = 0; i < route.ContainerTour.Count; i++)

            {
                if (route.ContainerTour[i].Id == wastecontainer.Id)
                {
                    savingsOfRemoval = DistanceMatrix[route.ContainerTour[i - 1].Id,
                    route.ContainerTour[i].Id] +
                    DistanceMatrix[route.ContainerTour[i].Id, route.ContainerTour[i +
                    1].Id] - DistanceMatrix[route.ContainerTour[i-1].Id,
                    route.ContainerTour[i + 1].Id];
```

```
253                 }
254             }
255             return savingsOfRemoval;
256         }
257 
258         public double CostsOfInsertion(Route route, WasteContainer wastecontainer)
259         {
260             double costsOfInsertion = 0;
261             int cheapestInsertionIndex = CheapestInsertion(route, wastecontainer);
262 
263             costsOfInsertion =
                DistanceMatrix[route.ContainerTour[cheapestInsertionIndex - 1].Id,
                wastecontainer.Id] + DistanceMatrix[wastecontainer.Id,
                route.ContainerTour[cheapestInsertionIndex].Id] -
                DistanceMatrix[route.ContainerTour[cheapestInsertionIndex - 1].Id,
                route.ContainerTour[cheapestInsertionIndex].Id];
264 
265             return costsOfInsertion;
266         }
267         //Search for route with lowest total fill level except route with index 0,
            because for this route no move is possi
268         public int SmallestRoute(List<Route> nearestNeighborRoutes)
269         {
270             int smallestRoute = int.MaxValue;
271             int smallestRouteIndex = 0;
272             List<int> routeIDs = new List<int>();
273 
274             foreach(Route r in nearestNeighborRoutes)
275             {
276                 if (r.GetTourFillLevel() > 0)
277                 {
278                     routeIDs.Add(r.ID);
279                 }
280             }
281 
282             foreach (Route route in nearestNeighborRoutes)
283             {
284                 int currRouteLength = route.GetTourFillLevel();
285                 if (currRouteLength > 0 && currRouteLength < smallestRoute &&
                    route.ID != routeIDs[0])
286                 {
287                     smallestRoute = currRouteLength;
288                     smallestRouteIndex = route.ID;
289                 }
290             }
291 
292             return smallestRouteIndex;
293         }
294 
295         public int CheapestInsertion(Route route, WasteContainer wasteContainer)
296         {
297             double smallestInsertion = double.MaxValue;
298             int smallestInsertionIndex = 0;
299 
300             for(int i = 0; i < route.ContainerTour.Count-1; i++)
301             {
302                 double sum = route.DistanceMatrix[route.ContainerTour[i].Id,
                    route.ContainerTour[i + 1].Id];
303                 if (sum < smallestInsertion)
304                 {
305                     smallestInsertion = sum;
306                     smallestInsertionIndex = i+1;
307                 }
308             }
309             return smallestInsertionIndex;
310         }
311 
312         public double CalculateCostsOfTour(List<Route> routes)
313         {
314             double costs = 0;
315 
316             foreach (Route route in routes)
317             {
318                 route.CalculateTourCosts();
319                 costs += route.TotalTourCosts;
320             }
321 
322             return costs;
323         }
324         public int DetermineNeighborhoodSearchIterations(List<Route> routeList)
325         {
326             int iterations = 0;
327             foreach (Route route in routeList)
328             {
329                 if(route.TotalTourCosts > 0 )
330                     iterations++;
331             }
332             return iterations;
333         }
```

```
334         public List<Route> CreateNewRouteList(List<Route> routeList)
335         {
336             List<Route> neighborhoodRoutes = new List<Route>();
337 
338 
339             foreach (Route route in routeList)
340             {
341                 int id = route.ID;
342                 double[,] distance = route.DistanceMatrix;
343                 List<WasteContainer> list = new List<WasteContainer>();
344 
345                 foreach (WasteContainer w in route.ContainerTour)
346                 {
347                     list.Add(w);
348                 }
349 
350                 Route r = new Route(id, list, distance);
351                 neighborhoodRoutes.Add(r);
352             }
353             return neighborhoodRoutes;
354         }
355     }
356 }
```