Faculty of Science and Technology
Department of Physics and Technology

Efficient Fuel Consumption Minimization for Green Vehicle Routing Problems using a Hybrid Neural Network-Optimization Algorithm

Astrid Fossum

EOM-3901 Master's thesis in energy, climate and environment - June 2023

“All models are wrong, but some are useful”
–George E. P. Box

Abstract

Efficient routing optimization yields benefits that extend beyond mere financial gains. In this thesis, we present a methodology that utilizes a graph convolutional neural network to facilitate the development of energy-efficient waste collection routes. Our approach focuses on a Waste company in Tromsø, Remiks, and uses real-life datasets, ensuring practicability and ease of implementation. In particular, we extend the DPDP algorithm introduced by Kool et al. (2021) [1] to minimize fuel consumption and devise routes that account for the impact of elevation and real road distance traveled. Our findings shed light on the potential advantages and enhancements these optimized routes can offer Remiks, including improved effectiveness and cost savings. Additionally, we identify key areas for future research and development.

Acknowledgements

I would like to express my heartfelt gratitude to my supervisors, Chiara Bordin and Michael Kampffmeyer, for their invaluable advice, support, and insightful discussions throughout my thesis journey.

Additionally, I am grateful for the collaboration with Remiks, the company I have had the privilege to work with. Their support and practical insights have been instrumental in ensuring the relevance and applicability of my research findings.

I would like to extend my sincere appreciation to my fellow classmates, whose companionship has been a constant source of encouragement and inspiration throughout these years. The pleasant memories we shared have truly enriched my academic experience.

Further, I want to thank my beloved Steinar for his support and encouragement throughout the ups and downs of this thesis undertaking, and my family for their unwavering love.

To all those mentioned above, I extend my heartfelt appreciation for being an integral part of my academic journey.

Contents

List of Figures

List of Tables

Abbreviations

CO2	Carbon Dioxide
CPF	Corner-Point Feasible
CVRP	Capacited Vehicle Routing Problem
GHG	Green House Gas
GNN	Graph Neural Network
ILP	Integer Linear Programming
ML	Machine Learning
MIP	Mixed Integer Programming
NN	Neural Network
OR	Operational Research
OP	Orienteering Problem
TSP	Travelling Salesman Problem
VRP	Vehicle Routing Problem
DP	Dynamic Programming
DPDP	Deep Policy Dynamic Programming
CNN	Convolutional Neural Network
SGD	Stochastic Gradient Descent

LP	Linear Programming
GVRP	Green Vehicle Routing Problem
TBL	Triple Bottom Line
ML	Machine Learning
HGS	Hybrid Genetic Search

1 Introduction

Efficient routing optimization has far-reaching benefits, extending beyond financial gains. One key advantage is the reduction of driving time. Optimization techniques aim to minimize travel distance and duration by strategically planning vehicle routes, enabling businesses to complete deliveries or services more time-effectively [2]. This enhances customer satisfaction through timely deliveries, reduces overall resource usage, and increases operational productivity. Such improvements align with sustainability efforts and the growing emphasis on environmentally friendly practices.

This thesis focuses on green vehicle routing in the context of waste management. The Vehicle Routing Problem (VRP) [2] is a widely recognized challenge in operational research. It revolves around determining the most efficient route for a group of vehicles to deliver goods or services to a specific group of customers. The VRP is a challenging problem with numerous real-world applications, such as transportation and delivery services, waste collection, and emergency response planning.

In recent decades, optimization techniques have been extensively used to solve complex problems like the VRP [2, 3, 4]. While exact and heuristic algorithms have made significant advancements in the VRP domain, there has been a growing interest in learning-based algorithms [5, 6]. Learning-based algorithms can learn optimization strategies from training data, distinguishing them from traditional exact and heuristic algorithms designed to solve predefined problems [7].

Solving the Vehicle Routing Problem (VRP) and its variants is known to be NP-hard, which implies that finding an exact solution would require exponential time since no polynomial-time algorithm exists [2]. Due to the typically large scale of real-time VRPs, heuristics, and learning-based methods are often considered suitable [8]. These algorithms provide solutions that may not be globally optimal but are highly practical and effective in real-world scenarios [9].

The minimization of fuel consumption has recently received significant interest in literature [10, 11, 12]. Another emerging approach is the combination of neural networks, and Operational Research (OR) [13]. State-of-the-art learning-driven methods for solving the Travelling Salesperson Problem (TSP) [14] can perform close to the exact solver when trained on trivial problem sizes [15]. However, they cannot generalize to data structures they have not been optimized for. Joshi et al. (2022) [15] provide the first principled investigation of zero-shot generalization, where the model is trained on trivial problem sizes of example data and generalizes to larger problems [15] or more complex data [16].

1.1 Context

This work was conducted in collaboration with Remiks Miljøpark AS, a waste collection company located in Tromsø, Norway. Remiks serves over 20,000 households in Tromsø and Karlsøy municipalities and primarily engages in waste collection, sorting, and disposal activities.

Remiks is a limited liability company owned by Tromsø and Karlsøy municipality. The concern has about 120 employees, and the head office is located north on the Tromsø island [17].

Remiks places great emphasis on environmental consciousness as one of its core values. This work focuses on the company's environmental consciousness and aims to generate robust waste collection routes that can adapt to unforeseen events, ensuring reliable and efficient operations.

1.2 Problem definition

In today's modern cities, waste collection is an essential logistic activity that must be carried out efficiently. To ensure optimal performance and timely delivery of services, companies like Remiks rely heavily on the experience and expertise of their employees. However, as the industry evolves, there is a need to find innovative ways to enhance waste collection operations.

One such enhancement is to utilize optimization methods that generate energy-efficient routes. By leveraging technology and data analysis, waste collection companies can optimize their routes, reduce fuel consumption, and minimize their carbon footprint.

While most papers in the literature focus on distance when optimizing VRPs [16, 18, 19], Tromsø and Karlsøy municipalities are characterized by tall mountains, islands, steep residential areas, and dispersed settlements. This intricate landscape poses challenges for route optimization, rendering distance-based approaches ineffective in achieving optimal results. To facilitate sustainable solutions, we hypothesize that the landscape and topology need to be taken into account in order to lower the fuel consumption of the Remiks vehicles.

1.3 Contributions

We propose to solve the VRP in this challenging setting by extending the DPDP algorithm introduced by Kool et al. (2021) [1] to minimize fuel consumption. We evaluate the method on the real-world data provided by Remiks. Our primary objective is to minimize the total fuel consumption of the vehicle rather than focusing on the distance. As training the model from scratch using the provided Remiks data would be time-consuming and require labels, we transfer the learned policies from Joshi et al. (2019) [20] and Kool et al. (2021) [1], inspired by Joshi et al. (2022) [15].

In this project, we aim to:

1. Incorporate a formula for calculating fuel consumption using real distances and elevation matrices obtained through the Google Maps commercial API [21].

2. Extend the DPDP algorithm introduced by Kool et al. (2021) [1] to minimize fuel consumption.

3. Evaluate and compare the obtained routes with the routes currently employed by Remiks.

1.4 Outline

The rest of this thesis is outlined as follows:

Part I - Background In Part I, we create a foundation for the approach proposed in Part II. In Chapter 2, "Graph-Based Operational Research," the TSP and VRP will be formally introduced. Also, we will present some commonly used solution methodologies for solving optimization problems using OR techniques. Chapter 3, "Fundamentals of Neural Networks: Building Blocks and Key Concepts," provides the foundations of fully connected neural networks. In Chapter 4, "Graph Convolutional Neural Networks," we move into Graph Convolutional Networks and their characteristics. Chapter 5, "Green Vehicle Routing Problem," introduces the Green VRP and describes how fuel consumption and environmental considerations are accounted for in the literature.

Part II - Method and data In Part II, we present the data and the method utilized for the experiments presented in Part III. Chapter 6 introduced the proposed method for measuring fuel consumption for Waste vehicles. Further, this method is used to optimize the routes of Remiks according to the DPDP framework. Chapter 7, "Data Gathering and Augmentations", describes the dataset used for the optimization and the parameters used.

Part III - Results and conclusion Part III consists of results, discussion, and conclusion. Chapter 8, "Results", displays the obtained results of the proposed algorithm. Chapter 9, "Discussion and Future Work" evaluates and discusses the results. Lastly, Chapter 10, "Concluding Remarks," summarizes this thesis's contributions and reflects on future directions.

Part I

Background

2 Graph-Based Operational Research

Graphs provide a powerful framework for modeling and analyzing complex systems in operational research. By representing entities as nodes and their relationships as edges, graph-based approaches enable researchers and practitioners to gain valuable insights into decision-making processes, resource allocation, and optimization problems. This chapter explores the intersection of graphs and operational research, highlighting the potential of graph theory in addressing operational challenges. Parts of this Chapter are based on initial research performed during the author's preliminary project [22].

Operational research is a broad field that encompasses research on operations in all kinds of organizations. Using mathematical and statistical methods, operational research can help organizations make better decisions, allocate resources more efficiently, and improve overall performance [23]. Effective routing and resource utilization are essential in order to minimize fuel consumption and greenhouse gas emissions [24].

Graph theory, as a branch of mathematics, offers a versatile toolset for capturing and analyzing intricate relationships, dependencies, and interactions between components within operational systems [25].

2.1 Operational research

Operational research provides valuable insights into the object being modeled, enabling experimentation and trial without the risk of real-world consequences. Algebraic symbolism is frequently employed in operational research to construct models that reflect the internal relationships within an object or use case.

When using mathematical models to model real-world relationships, it is critical to utilize the mathematical model as a means to gain decision insight, rather than relying solely on it to make decisions [26]. Additionally, it is essential to verify and evaluate the simplifications and calculations that are being used.

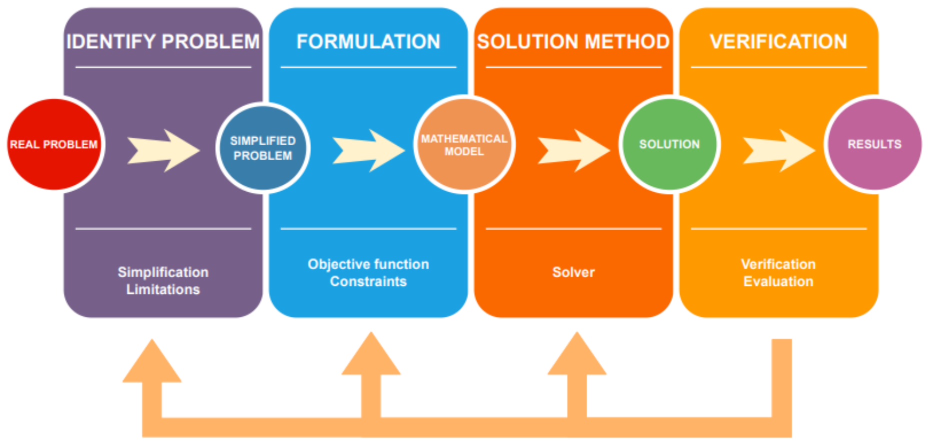


Figure 2.1: Flow chart of a mathematical model. The approach is based on the methodology described by Greefrath and Vortholer [27].

Figure 2.1 depicts the approach utilized to simplify Remiks' fuel minimization Waste Vehicle Routing Problem into an optimization problem. The methodology follows the framework described by Greefrath and Vortholter (2016) [27].

Mathematical models generally consist of several components, including input data, an objective function, constraints, and decision variables [27]. Input data serves as the foundation of information upon which decisions are based and is predetermined. The objective function defines the aim of the optimization and determines whether decision variables $(x_1, x_2, ..., x_n)$ should be minimized or maximized. Examples of objective functions include maximizing profit, minimizing cost or maximizing net present value. Constraints, expressed as equality or inequality equations, place limitations on decision variables and establish the relationship between supply and available resources. For example, a vehicle may have an upper weight limit that restricts how much it can load before returning to the depot. The goal of optimization problems is to identify optimal values for decision variables that maximize or minimize the objective function [23]. Figure 2.2 illustrates the interplay between input data, the objective function, constraints, and decision variables.

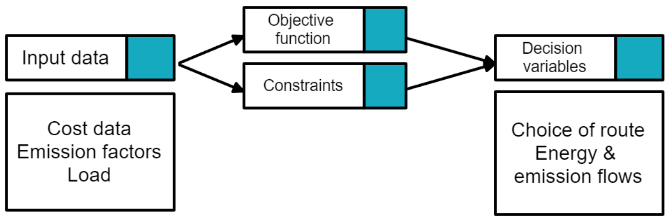


Figure 2.2: Main inputs in a mathematical model. The input data is fed into the model. The constraints are applied to the input data in order to find the decision variables that optimize the objective function.

2.1.1 Linear and Mixed Integer Programming

Linear programming is a mathematical approach utilized to optimize the distribution of resources among competing activities. Its objective is to discover an optimal solution by constructing a mathematical model that represents the problem at hand. All mathematical functions in a linear programming problem must be linear in nature [28]. One widely employed technique for solving linear programming problems is the Simplex algorithm [29]. This algorithm focuses on determining solutions situated at the corner points of the feasible region, known as Corner-Point Feasibles (CPFs).

The Simplex algorithm initiates from an arbitrary vertex and iteratively examines whether any neighboring vertex can enhance the optimal solution. If an admissible vertex produces a better value for the objective function, the solution adopts that value. In the event that no adjacent vertices can improve the objective function, the algorithm concludes that the current vertex represents the optimal solution [28].

When a problem requires only a subset of variables to be integers, it is referred to as a Mixed Integer Programming (MIP). This technique is commonly employed for problems involving interdependent binary decisions that can be represented by binary variables [23]. The branch-and-bound method [30] is often utilized to solve binary integer programming problems. This method involves decomposing the original problem into smaller, more manageable subproblems, a process known as branching [31]. The subproblems are then solved using a relaxation of the problem. The most frequently used relaxation technique for MIP involves specifying certain variables as integers and applying the Simplex method to solve the linear programming relaxation of the problem [23].

A subproblem is dismissed from further consideration if [23]:

- The Linear Programming (LP) relaxation has no feasible solutions,
- The solution of the subproblem is worse than the current best solution,
- The current best solution gets replaced if the proposed solution is superior.

The iterations end when there are no remaining subproblems that have not been explored. Then, the current best solution is optimal.

2.1.2 Dynamic programming

Dynamic programming is a powerful mathematical technique used to solve problems that require a series of interconnected decisions and that can be divided into overlapping subproblems. This approach uses a bottom-up strategy to address subproblems by initially solving equations with the smallest mathematical value and then working upwards. The solutions obtained from the subproblems can then be utilized to solve larger subproblems. Richard Bellman developed dynamic programming in the 1950s [32].

Dynamic programming involves determining necessary decisions at each time step based on the current state of the system. The key features that define dynamic programming include dividing the problem into stages, associating each stage with a set of possible system conditions, and transforming the current stage into a stage associated with the next stage based on policy decisions [23]. The stages are visualized in Figure 2.3. The goal of Dynamic Programming is to find an optimal policy for the entire problem.

Dynamic programming provides a policy description of what to do under every possible circumstance. Given the current state, the optimal policy for the remaining stages is independent of the policy decisions made in previous stages. The solution procedure starts by finding the optimal policy for the last stage and then moves backward, stage by stage, to find the optimal policy for each preceding stage. A recursive relationship identifies the optimal policy for each stage.

The Traveling Salesman Problem (TSP) can be considered a multistage decision problem. For a tour to be optimal, it must have a minimum total length. Let

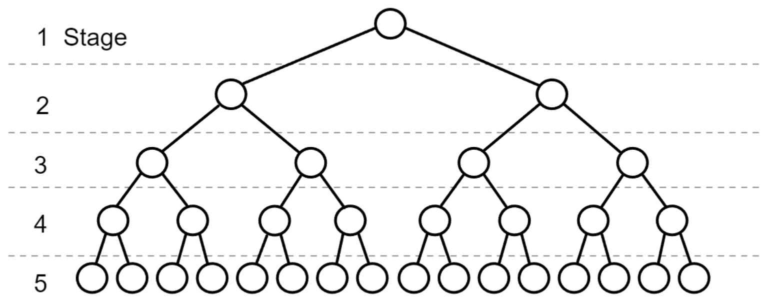


Figure 2.3: Problem suitable for dynamic programming. Starting from the upper node, the problem can easily be divided into smaller subproblems based on the decision made at each stage.

us denote $f(i : j_1, j_2, ..., j_k)$ as the length of the shortest tour path starting from location 0 and ending at location i, while ensuring that nodes $j_1, j_2, ..., j_k$ are visited exactly once along the path. We define d_{ij} as the distance between the ith and jth cities. Hence, the tour of the minimum length can be defined as:

$$f\left(i; j_1, j_2, \cdots, j_k\right) = \min_{1 \leqq m \leqq k} \left\{d_{ij_m} + f\left(i; j_1, j_2, \cdots, j_{m-1}, j_{m+1}, \cdots, j_k\right)\right\} \tag{2.1}$$

This equation can be generalized into the equation:

$$f(i; j) = d_{ij} + d_{j0}. \tag{2.2}$$

Using Equation 2.2, we obtain $f(i; j_1, j_2, \cdots, j_n)$, and through Equation 6.1, we get $f(i; j_1, j_2)$ until $f(0; j_1, j_2, \cdots j_n)$ is obtained. The sequence of values m that results in the minimum value of Equation 6.2 provide the shortest path.

Dynamic programming's policy is to optimize the score function, which is expressed as $score = heat + potential$. The heated term represents the cost of the edges included in the solution, while the potential term estimates the cost of the unvisited nodes based on the remaining edges. We initiate our calculation at the bottom stage of the graph where all information is readily available. From there, we can systematically compute the heat and potential values for the entire graph [23].

While dynamic programming is a highly effective technique for solving problems that involve interconnected decisions, it may suffer from poor scalability in certain circumstances. This is because the number of subproblems that need to be solved rapidly increases as the size of the problem grows. As a result, the computational time required to solve the problem may become prohibitively large, rendering dynamic programming impractical [23].

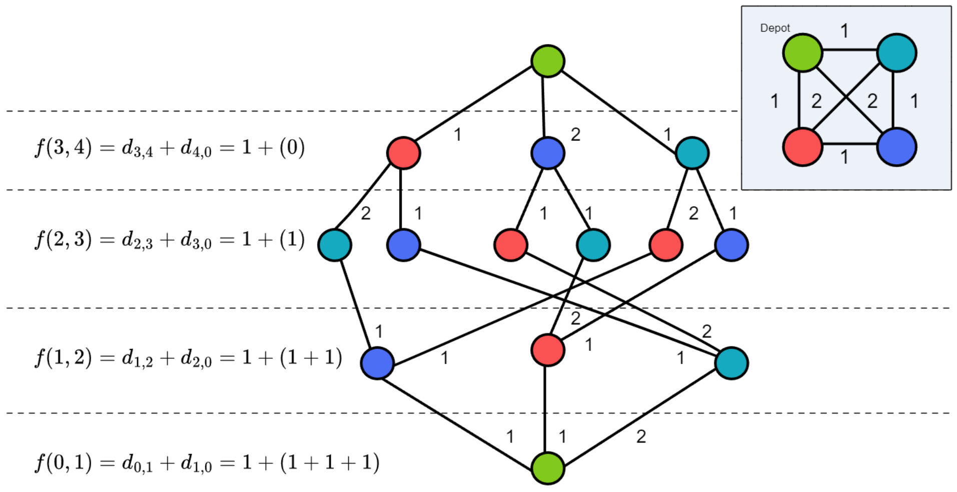

Depot
$f(3,4) = d_{3,4} + d_{4,0} = 1 + (0)$
$f(2,3) = d_{2,3} + d_{3,0} = 1 + (1)$
$f(1,2) = d_{1,2} + d_{2,0} = 1 + (1+1)$
$f(0,1) = d_{0,1} + d_{1,0} = 1 + (1+1+1)$

Figure 2.4: Visualization of dynamic programming. $d_{i,j}$ is the cost of going from the current node to the next. $d_{j,0}$ represents the lowest cost of going from the net node to the depot, i.e., the potential.

In Chapter 6, a Graph Convolutional Neural Network will be utilized in order to simplify a dynamic programming problem.

2.2 The Traveling Salesman Problem

The TSP is a special case of the VRP where only one vehicle is available [23]. It is a classic problem in operational research that greatly benefits from graph theory. The TSP involves finding the shortest possible route that allows a salesperson to visit a set of cities and return to the starting point while visiting each city exactly once [26]. The problem is visualized in Figure 2.5. This seemingly simple problem is known to be NP-hard, meaning that as the number of cities increases, the computational complexity grows exponentially [2].

The problem is defined on a graph $G = (N, A)$, with a cost c_{ij} for every arc $(i, j)\epsilon A$,

$$\delta_{i,j} = \begin{cases} 1 & \text{if the tour goes from } i \text{ to } j \text{ direct} \\ 0 & \text{otherwise.} \end{cases}$$

Williams (2013) book *Model Building in Mathematical Programming* [26] used the following notations to describe the TSP:

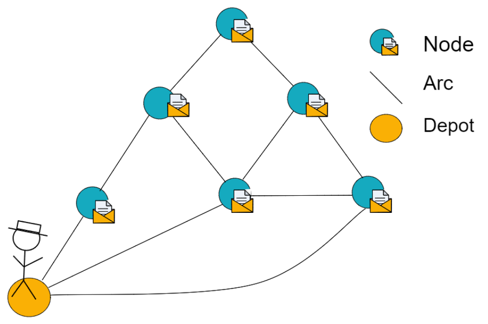


Figure 2.5: traveling salesman problem. The salesman starts from home and has to visit all cities before returning to the starting point. The optimal route is the order of cities that cover the shortest total distance.

The objective of a TSP is to minimize the objective function,

$$min. \sum_{i,j} c_{i,j}\delta_{i,j},$$

where $c_{i,j}$ is the cost between node i and j. Three conditions have to be satisfied in a traditional TSP [26]:

1. Exactly one city must be visited immediately after city i,

$$\sum_{j=0,i\neq j} \delta_{ij} = 1, i = 0, 1, ..., n. \tag{2.3}$$

2. Exactly one city must be visited immediately before city j,

$$\sum_{i=0,i\neq j} \delta_{ij} = 1, j = 0, 1, ..., n. \tag{2.4}$$

3. The optimal route can not contain sub-tours. A sub-tour is a cycle that does not visit all nodes,

$$\sum_{(i,j)C} \delta_{ij} \leq (|C| - 1)\forall C \triangleright \text{C is a sub-tour in G.} \tag{2.5}$$

In order to tune the problem for a specific real-world problem, additional constraints can be added, or the objective function can be modified.

2.3 The Vehicle Routing Problem

The objective of the VRP is to determine the optimal delivery routes for a fleet of vehicles that visit a set of dispersed customers while meeting various constraints and minimizing costs [33]. Due to the diverse range of constraints that may be imposed, the VRP plays a critical role in managing various delivery services and arises in a wide range of different forms.

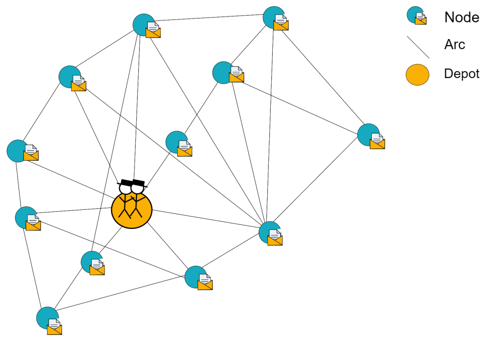


Figure 2.6: The Vehicle Routing Problem. The salesmen start from the depot and have to visit all cities before returning to the starting point. The optimal route is the order of cities that cover the shortest total distance.

In a VRP, the vehicles generally have a fixed capacity expressed as the maximum weight they can carry and a cost associated with their utilization per distance, time, or route to prevent the model from employing an infinite number of vehicles [2].

The Capacited Vehicle Routing Problem (CVRP) is the simplest case of the VRP where only the capacity constraints are present, introduced by Dantzig and Ramser in 1959 [34]. The number of vehicles is fixed and decided in advance. Each node has a demand, and each vehicle has a maximum capacity. The total demand of a route can not exceed the vehicle's total capacity [35].

Typical objectives for VRP are minimizing the global transportation cost or minimizing the number of vehicles required to serve all customers. In the case of Figure 2.6, an objective could be to minimize the distance traveled to visit all nodes [36]:

$$min \sum_{(i,j)\epsilon A} c_{ij} \sum_{l\epsilon K} x_{ij} \tag{2.6}$$

Where:

$G = (V, A)$ is a directed graph,
$V = 0, ..., n$ is the set of nodes,
$A = (i, j)|i, j\epsilon V, i \neq j$ is the set of arcs,
c_{ij} is the cost associated with arc(i, j)ϵA, for instance distance,
$x_{ij}\epsilon(0, 1) = 1$, only if a vehicle uses the arc (i,j)ϵA,

Typical constraints for a traditional VRP are the following [2]:

1. Each customer node in a route is connected to two other nodes,

$$\sum_{j\epsilon\delta+(i)} = 1, \sum_{i\epsilon\delta-(j)} = 1. \tag{2.7}$$

2. Exactly K routes are constructed,

$$\sum_{j\epsilon\delta+(0)} x_{0j} = |K|. \tag{2.8}$$

3. The capacity of the vehicles is not exceeded,

$$\sum_{(i,j)\epsilon\delta+(S)} x_{ij} \geq r(S). \tag{2.9}$$

In a VRP, a given number of vehicles will start from the depot and collectively visit all nodes at the minimum total cost.

2.4 Solution methodologies

Solving the Vehicle Routing Problem (VRP) and its variants is known to be NP-hard, which implies that finding an exact solution would require exponential time since no polynomial-time algorithm exists [2]. To tackle this challenge, exact methods approach the VRP as integer or mixed-integer programs to identify near-optimal solutions. However, due to the typically large scale of real-time VRPs, heuristic-based methods are often considered more suitable [8]. More than 70% of the solution methods found in the literature are based on meta-heuristics, which offer the advantage of escaping from local minimums but cannot guarantee optimality [37, 38].

Three general types of solutions to VRP have been presented in the literature:

Exact algorithms An exact algorithm can find the optimal solution, but its heavy mathematical programming makes it only suitable for small-sized problems [12]. Exact algorithms are typically based on Integer Linear Programming (ILP), dynamic programming, or branch-and-bound programming [39]. Linear programming involves optimizing a linear function subject to various constraints. Dynamic programming is a useful mathematical technique for making a sequence of interconnected decisions. Branch-and-bound programming is frequently used to solve MIP [31].

Heuristic algorithms A classical heuristic algorithm is an experience-based technique for finding solutions through a series of iterative processes. The improvement steps in the approach always descend [39], meaning that the following solution is consistently better than the previous one. It requires a relatively short execution time but does not guarantee the solution's optimality [12].

Metaheuristics Metaheuristics are random-based algorithms for exploring possible solutions in a large range [12]. Metaheuristics consist of first generating a family R' of feasible routes and then solving the formulation over R' rather than the full set of R. The success of the algorithm depends on the quality of the generated routes. In contrast to exact and heuristic algorithms, meta-heuristics allow for non-improving and infeasible intermediate solutions [39].

A great variety of metaheuristic schemes have been put forward. They can broadly be classified into three categories:

- A **local search heuristic** forms an initial solution s_0 (which may be infeasible) and moves at each iteration t from a solution s_t of value $f(s_t)$ to another solution located in the neighborhood $N(s_t)$ on s_t. The neighborhood $N(s_t)$ consists of all solutions that can be reached from s_t by applying a given type of transformation, for example, relocating a vertex from its current route into another route. The search ends with the best-known solution s' after a stopping criterion has been satisfied [39].

- **Genetic algorithms** evolve a population of solutions through mutation. In the context of VRP, The Hybrid Genetic Algorithm [40] has been proposed to transform the routing problem into a TSP by removing the factor of multiple vehicles. The problem is optimized, then the VRP solution is reconstructed at the end of the process [39]. Rochat and Taillard (1995) [41] presented an algorithm that takes the best solutions found using the Tabu search algorithm [42] and recombines them using crossovers and local search to provide even better solutions [39].

- Ant colony optimization and neural networks are two examples of **learning mechanisms**. Ant colony optimization heuristics attempt to mimic the behavior of ants who detect paths containing pheromones and strengthen them with their pheromones [23]. In meta-heuristics, the pheromones represent the system's memory and correspond to edges often appearing in suitable solutions. Thus, the algorithm remembers good edges and is more likely to include them in a solution [39]. Neural Networks (NN) were applied for operational research for the first time by Hopfield and Tank in 1985 [43], in the form of a Hopfield network, to solve a simple TSP problem. In recent times, neural networks for optimization have gained significant prominence, and techniques such as Attention [13], LSTM [44], and Reinforcement Learning [45] have been suggested. Chapter 4 will introduce GNN and how they can be employed to solve routing problems.

Metaheuristic optimization methods are a good choice for solving VRP for several reasons. These complex problems have large solution spaces, making exact methods inefficient. Metaheuristic methods offer efficient search strategies to navigate the solution space and provide near-optimal solutions for large-scale instances. They also handle nonlinear relationships and constraints in VRP and TSP and balance solution quality and computational efficiency [46]. Additionally, metaheuristic methods are adaptable to real-world dynamics, can handle problem variants and extensions, and are practical for real-world applications.

2.5 Abstractions and Simplifications

Mathematical optimization is a robust tool that effectively tackles intricate problems [38]. However, it is essential to acknowledge that optimization models are mathematical abstractions of real-world problems. They assist decision-making by offering a systematic framework to allocate resources, enhance efficiency, reduce costs, or attain desired objectives. It is important to understand that these models should be utilized as a basis for decision-making rather than treated as absolute truths.

Two primary approaches have been employed in the literature to simplify real-world problems into OR frameworks: the top-down approach and the bottom-up approach [47].

The top-down approach involves initiating the modeling process with a high-level understanding of the problem [23]. It entails identifying overall objectives, constraints, and key decision variables. The model is then decomposed into sub-problems, gradually refining the model by incorporating more detailed information. This approach emphasizes a global perspective and aims to capture the system's overall behavior. It is particularly useful for complex problems that require a holistic understanding.

In contrast, the bottom-up approach starts with detailed information about individual components or subsystems of the system. It models the specific interactions and constraints within each component and subsequently integrates them to form a comprehensive model [23]. This approach is beneficial when the system's behavior emerges from the interactions of its individual parts. It allows for a more detailed analysis of local behaviors and their impact on the overall system.

Both approaches have their strengths and weaknesses. The top-down approach is beneficial for comprehending the overall system dynamics and capturing global patterns and behaviors. However, it may oversimplify or overlook local details that could impact the system's behavior. The bottom-up approach is valuable for capturing detailed interactions and behaviors, but it may struggle to incorporate the broader system-level constraints or objectives.

For the simplification of the Remiks problem, the top-down approach has been used. This approach offers insights into how Remiks operate and identifies their needs.

Metrics As a decision maker, evaluating and verifying the solution provided by an operational research algorithm is essential. In their work, Vidal and Laporte (2019) [48] identified seven key categories of metrics that should be considered to generate an optimized route feasible for practical applications.

The following is a summary of the seven metrics:

Profitability Cost optimization or profit maximization is typically the primary objective of most VRP studies. Profit can be optimized to maximize fleet utilization or the efficiency of the performed task.

Service quality High-quality service is essential for maintaining clients. This involves ensuring that vehicles arrive on time if there are time window restrictions. To ensure service quality, it is important to ensure that the routes are robust to unforeseen events.

Equity Workload balance for vehicles and employees is essential to maintain employee satisfaction and prevent resource utilization bottlenecks. In a VRP, work equity can be implemented by ensuring that all vehicles perform the same amount of work or use the same amount of time.

Consistency Drivers perform better and drive more safely if they are familiar with their route. In the case of Remiks, all households must be visited once a week on the same day. The Remiks drivers state that their customers expect their bins to be emptied simultaneously. Hence, the consistency and reliability of the routes are important.

Simplicity Route plans should be easy and intuitive, with geographically distinct routes facilitating coordination. Rossit et al. (2019) emphasize that compact, non-overlapping, and incomplex routes are desirable to generate visually attractive routes [49]. Further, they state that visually attractive routes require less explanation and training for the drivers to use and enhance the acceptance among drivers and customers.

Reliability It is crucial to account for uncertainties and ensure that the chosen path remains viable even in unforeseen circumstances, such as unexpected traffic congestion or road closures.

Externalities Traditional cost-minimization objectives based on distance or time may not lead to minimal emissions or consumption, so other metrics must be considered to achieve a "green" VRP. Other examples of externalities to consider are safety risks and traffic.

The evaluation metrics indicate that achieving the highest cost-effectiveness alone may not always suffice for an optimal route plan. Organizations strive to optimize operational robustness [50], minimize overtime and delays [51], and accomplish various other objectives. In many scenarios, it is necessary to address multiple objectives concurrently, thereby tackling multiple objective problems.

One way of doing this is to incorporate one of the objectives as constraints. This would be desirable for a routing problem where one wishes to minimize driving distance as long as the fuel consumption does not exceed a specific number. Another way of tackling multiple objectives is to take a suitable linear combination of all the objective functions and optimize that. This approach would require relative weighting between the parameters [23].

2.6 Challenges

As mentioned in the previous section, OR models rely on simplifications, and excessive simplifications can lead to inaccuracies in representing reality.

The mathematical complexity of the VRP poses another challenge for OR. The VRP is classified as an NP-hard problem, implying that as the problem size increases, finding the exact solution becomes exponentially time-consuming [23]. For large-scale VRP instances, it may be impractical to obtain the optimal solution.

Considering these challenges, neural networks offer advantages in solving optimization problems. They excel in capturing complex and nonlinear relationships among variables, making them well-suited for optimization problems with high-dimensional input spaces and numerous constraints [52]. Furthermore, neural networks can adapt and generalize to new problem instances, learn from data to provide accurate and efficient solutions, and leverage parallel computing architectures for faster optimization.

In the context of Remiks route optimization, a neural network approach is employed to constrain the search space for a dynamic programming optimization algorithm. Chapter 3 will provide a brief introduction to neural networks and their mechanisms.

3 Fundamentals of Neural Networks: Building Blocks and Key Concepts

Machine learning aims to develop algorithms that optimize performance based on example data or prior knowledge, using statistical methods to identify relationships and patterns in training data [7].

Neural networks distinguish themselves from traditional machine learning algorithms by their capability to extract high-level features from data with minimal human intervention. During training, neural networks autonomously discover dependencies, allowing for a more automated and efficient learning process [53].

This chapter will describe the foundation of neural networks and provide an introduction to relevant mechanisms and concepts.

3.1 Fundamentals of Neural networks

Neural networks exhibit diverse architectures tailored to specific problem domains. In a feedforward neural network, neurons are organized into layers, with each layer receiving input from the preceding layer and generating output for the subsequent layer. The input layer serves as the initial layer, followed by hidden layers that perform computations to facilitate accurate predictions, ultimately leading to the output layer.

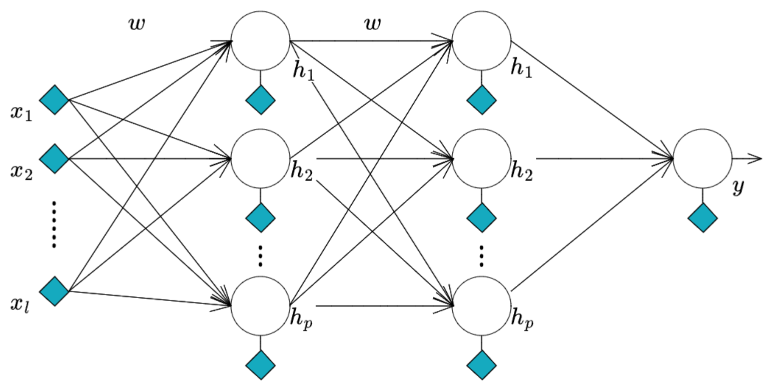


Figure 3.1: An example of a multilayer perception with two hidden layers, *x* denotes the input layer, and *h* denotes the hidden layers. The predictions or classifications happen at the layer *y*.

The perceptron is a simple neural network structure that can be used for classification and regression [53]. The structure of the perceptron is important, as it lays the foundation for the structure of deeper neural networks. Figure 3.1 represents a multilayer perceptron with two hidden layers.

The input **x** is applied an *activation function* in the forward direction to calculate the hidden units *h*. Each hidden unit in the perceptron is a perceptron by itself and applies a nonlinear activation to its weighted sum. *Activation functions* will be described further in Section 3.1.3.

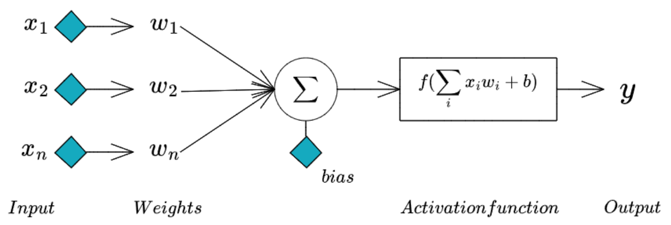


Figure 3.2: Visualization of the neuron with n inputs. The neuron receives input from the previous layer, calculates the sum of the signals multiplied by their respective weights, and then applies the *activation function* to the resulting value.

Figure 3.2 visualize what is happening within an arbitrary neuron in the architecture of Figure 3.1. Multilayer perceptrons can implement nonlinear discriminants and approximate nonlinear functions of the input [54].

Layers in neural networks are groups of interconnected units. These layers are arranged in a sequential structure, with each layer being a function of the previous layer. In a fully connected layer, every feature in layer l directly affects every feature in layer $l + 1$.

In general, fully connected layers in a neural network are expressed as [54]:

$$\begin{aligned} \mathbf{h}^{(0)} &= x, \\ \mathbf{h}^{(1)} &= g^{(1)}(\mathbf{w}^{(1)T}\mathbf{y}^{(0)} + \mathbf{b}^{(1)}), \\ \mathbf{h}^{(2)} &= g^{(2)}(\mathbf{w}^{(2)T}\mathbf{y}^{(1)} + \mathbf{b}^{(2)}), \\ &\vdots \\ \mathbf{y}^{(l)} &= g^{(l)}(\mathbf{w}^{(l)T}\mathbf{y}^{(l-1)} + \mathbf{b}^{(l)}) \end{aligned}$$

Where,

$\mathbf{x} \in \mathbb{R}^{F_o \times 1}$ is the feature input to the model.

$l = 1, 2, 3, ...L$ is defined as the number of layers in the network.

$\mathbf{h}^{(l)} \in \mathbb{R}^{F_l \times 1}$ is the hidden state feature vector at the l-th layer.

$\mathbf{w}^{(l)} \in \mathbb{R}^{F_l \times F_{l-1}}$ is the weight matrix.

F_l is the dimension of the vectors and is equal to the number of neurons in each layer.

$\mathbf{y}$ represents the final prediction that is performed based on the previous calculations.

g represents the activation function that allows the neural network to model complex nonlinear dependencies. Activation functions will be described further in section 3.2.4.

Having described these parameters, we are ready to explore further how this structure can be used for training.

3.2 Training a perceptron

To evaluate the performance of a neural network, *objective functions* are employed. These functions take the network's prediction as input and generate a measure of the prediction's performance. Further, the network's weights and biases can be adjusted to optimize the value objective function, aiming to achieve the best attainable value.

In this section, the mechanisms that are used to perform these adjustments effectively are introduced.

3.2.1 Gradient Descent

One of the most commonly used methods for improving a network's weights and biases is *Gradient Descent* [54]. The starting point of the gradient descent algorithm is an arbitrary point on the objective function. From there, the steepness of the slope is measured using a tangent. The goal of the algorithm is to minimize the objective function or the error between the predicted and actual y-value [55]. When the weights and the biases are updated, the slope of the function should be gradually slighter until we arrive at a global or local minimum or maximum. Figure 3.3 visualize these parameters.

The learning rate is an important hyperparameter for gradient descent that specifies the size of the step taken during each parameter update [54]. The learning rate can be chosen through trial and error or by monitoring learning curves that plot the objective function over time.

Algorithm 1 describes how the weights are updated when optimizing using gradient descent.

The update function takes the gradients as input and adjusts the weights of the network. The basic iteration step is on the form

$$\theta_j^r(new) = \theta_j^r(old) + \Delta\theta_j^r \tag{3.1}$$

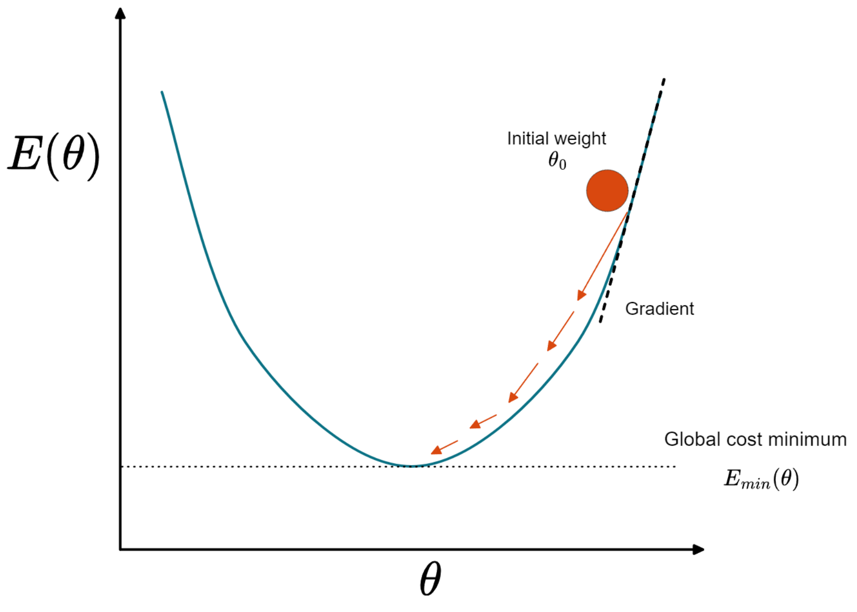


Figure 3.3: Visualization of gradient descent. θ_0 is the initial weight and the starting point on the optimization function. The arrows represent the step size μ. The objective of gradient descent is to reach the global cost minimum of the objective function.

Algorithm 1 Gradient descent with respect to parameter θ

```
**Input:** Initial parameter guess $\theta_0$ and a learning rate $\mu \in |\mathbb{R}|$

**while** *minimum is not reached* **do**
    $\Delta\theta_j^r = -\mu * \frac{\partial E}{\partial w_j^r},$        ▷ E is the objective function
    $\theta_j^r(new) = \theta_j^r(old) + \Delta\theta_j^r$
  **Result:** $\theta_{min} \leftarrow \theta_{i+1}$
```

where

$$\Delta\theta_j^r = -\mu \cdot \frac{\partial E}{\partial w_j^r}, \tag{3.2}$$

$\theta_j^r(old)$ is the current estimate of the unknown weights and $\Delta\theta_j^r$ is the corresponding correction to obtain the next estimate $\theta_j^r(new)$.

Now, a method for estimating $\Delta\theta_j^r$ is required.

3.2.2 Backpropagation

Gradient descent requires a learning rate and a gradient of error. While the learning rate often is decided through trial and error, the gradient is calculated using backpropagation [53]. Backpropagation leverages the layer structure of neural networks to calculate the gradients of the weights for each layer sequentially [54]. By utilizing the chain rule of differentiation [56], the weights can be updated layer by layer. The backpropagation algorithm facilitates the flow of information from the cost function, propagating it back through the network to calculate the gradient necessary for learning and performing gradient descent [53].

Figure 3.4 visualize the variables involved in backpropagation. v_j^r represents the weighted summation of the inputs to the jth neuron of the rth layer, and y_j^r represents the corresponding output after the *activation function*.

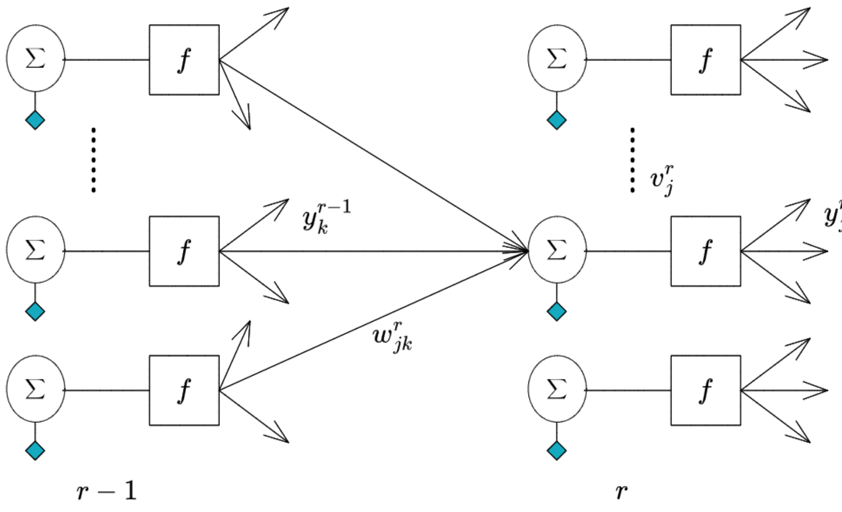


Figure 3.4: Visualiation of the variables involved in backpropagation. v_j^r is the weighted summation of the inputs to the jth neuron of the rth layer, and y_j^r is the corresponding output after the *activation function*.

The gradient of error, $\Delta\theta_j^r$, can be calculated for all layers using the formulas of Algorithm 2. The mathematical formulations will be further described below.

Algorithm 2 The Backpropagation Algorithm

```
Require: Init all weights with random small values.

  function FORWARD PROPAGATION(i,j)
      for all $x(i), i = 1, 2, ..., N$ do
          Compute all $v_j^r(i), y_j^r(i) = f(v_j^r(i)), \quad j = 1, 2, ..., k_r, \quad r = 1, 2, ..., L$

          Compute cost function for the current weight

          function BACKWARD PROPAGATION(i,j)
              for each $i = 1, 2, ..., N$ and $j = 1, 2, ..., k : l$ do
                  Compute $\delta_j^L(i)$
                  Compute $\delta_j^{r-1}(i)$ for $r = L, L-1, ..., 2$ and
                  $j = 1, 2, ..., k_r$

                  function UPDATE THE WEIGHTS(r,j)
                      for $r = 1, 2, ..., L$ and $j = 1, 2, ..., k_r$ do
                          $\theta_j^r(new) = \theta_j^r(old) + \Delta\theta_j^r$
                          $\Delta\theta_j^r = -\mu \sum_{i=1}^N \delta_j^r(i) y^{r-1}(i)$
```

From Equations 3.1 and 3.2, which describe the basic iteration step of back-propagation, we define all objective functions on the form

$$E = \sum_{i=1}^{N} \epsilon(i), \tag{3.3}$$

where ϵ is an approximated function dependent on $\hat{y}(i)$ and $y(i), i = 1, 2, ...N$. E is expressed as a sum of the N values that function ϵ takes for each of the training pairs $(y(i), x(i))$. $\epsilon(i)$ can, for instance, be the sum of squared errors of the output neurons.

By defining

$$\frac{\partial \epsilon(i)}{\partial v_j^L(i)} \equiv \delta_j^r(i) \tag{3.4}$$

Equation 3.2 become

$$\Delta\theta_j^r = -\mu \sum_{i=1}^{N} \delta_j^r(i) y^{r-1}(i), \tag{3.5}$$

and describes the change in the objective function for any differentiable objective function on the form (3.3).

Further, the backpropagation algorithm calculates the value of $\delta_j^r(i)$ by propagating backwards from $r = L$ and propagate backwards for the values $r = L-1, L-2, ..., 1$.

For the last layer where r = L, we have that:

1. $r = L$

$$\delta_j^L(i) = \frac{\partial \epsilon(i)}{\partial v_j^L(i)} \tag{3.6}$$

$$\epsilon(i) \equiv \frac{1}{2}\sum_{m=1}^{k_L} e_m^2(i) \equiv \frac{1}{2}\sum_{m=1}^{k_L}\left(f\left(v_m^L(i)\right) - y_m(i)\right)^2 \tag{3.7}$$

Hence

$$\delta_j^L(i) = e_j(i) f'\left(v_j^L(i)\right)$$

where f' is the derivative of $f(\cdot)$. In the last layer, the dependence of $\epsilon(i)$ on $v_j^L(i)$ is explicit, and the computation of the derivative is straightforward. For hidden layers, the computations of the derivatives need more elaboration [54].

For the previous layers, we have that:

2. $r < L$**.** Due to the successive dependence among the layers, the value $v_j^{r-1}(i)$ influences all $v_k^r(i), k = 1, 2, \ldots, k_r$, of the next layer. Employing chain rule in differentiation, we obtain

$$\frac{\partial \epsilon(i)}{\partial v_j^{r-1}(i)} = \sum_{k=1}^{k_r} \frac{\partial \epsilon(i)}{\partial v_k^r(i)} \frac{\partial v_k^r(i)}{\partial v_j^{r-1}(i)} = \delta_j^{r-1}(i) = \sum_{k=1}^{k_r} \delta_k^r(i) \frac{\partial v_k^r(i)}{\partial v_j^{r-1}(i)}, \tag{3.8}$$

From the definition of Equation 3.4

$$\frac{\partial v_k^r(i)}{\partial v_j^{r-1}(i)} = \frac{\partial\left[\sum_{m=0}^{k_{r-1}} \theta_{km}^r y_m^{r-1}(i)\right]}{\partial v_j^{r-1}(i)} \tag{3.9}$$

with

$$y_m^{r-1}(i) = f\left(v_m^{r-1}(i)\right) \tag{3.10}$$

Hence,

$$\frac{\partial v_k^r(i)}{\partial v_j^{r-1}(i)} = \theta_{kj}^r f'\left(v_j^{r-1}(i)\right) \tag{3.11}$$

From (3.10) and (3.8), the following results:

$$\delta_j^{r-1}(i) = \left[\sum_{k=1}^{k_r} \delta_k^r(i)\theta_{kj}^r\right] f'\left(v_j^{r-1}(i)\right) \tag{3.12}$$

which enables $\delta_j^{r-1}(i)$ to be expressed as

$$\delta_j^{r-1}(i) = e_j^{r-1}(i) f'\left(v_j^{r-1}(i)\right) \tag{3.13}$$

where

$$e_j^{r-1}(i) = \sum_{k=1}^{k_r} \delta_k^r(i)\theta_{kj}^r.$$

Now $\Delta\theta_j^r$ can be calculated using Equation 3.5.

To summarize, backpropagation involves calculating the gradients of the loss function concerning the network's weights and biases. These gradients indicate how each weight and bias should be adjusted to minimize the prediction error. By iteratively updating the weights and biases based on these gradients, the network learns from its mistakes and improves its accuracy through gradient descent.

3.2.3 The Vanishing and Exploding Gradient problems

The vanishing gradient problem is a common issue that arises during the training of deep neural networks. It occurs when the gradient values become very small during gradient descent. As a result, the weights of the early layers are updated very slowly or not at all, which can cause these layers to become ineffective in the learning process [57].

Similarly, the model's training becomes unstable when the gradient values become very large. This problem is called the exploding gradient problem.

To address these issues, several techniques have been developed, including the use of *activation functions* with higher gradients [58], proper initialization of network weights [54], normalization techniques such as *batch normalization* [54], and the integration of *skip connections* to allow gradients to bypass specific layers [57].

3.2.4 Activation function

Previously, activation functions were introduced as a function applied to the neurons in the hidden layer of a neural network. In this section, we will dive further into activation functions.

An activation function is a mathematical operation that acts on the activation level of a neuron to decide its activity. It takes the sum of the input signals and transforms it into an output value, which is then forwarded to the subsequent neural network layer. Acting as a transfer function, the activation function computes calculations within each neuron [58].

The activation function primarily aims to introduce nonlinearities to the neural network to model nonlinear relationships [54].

ReLU, Tanh, and Softmax are three commonly used activation functions in neural networks.

Sigmoid

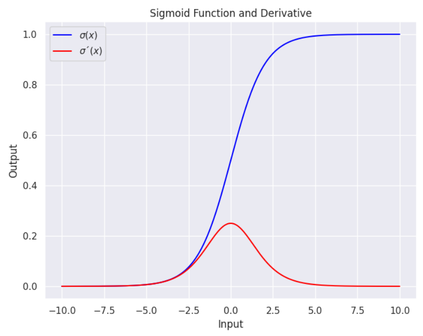


Figure 3.5: The sigmoid activation function.

The sigmoid activation function is widely used in neural networks, particularly for binary classification tasks. It maps any input value to a value between 0 and 1, providing a smooth and continuous transition from the minimum value to the maximum value. The sigmoid function is defined mathematically as

$$Sigmoid(x) = \frac{1}{1+e^{-x}}, \tag{3.14}$$

Where e is the mathematical constant known as Euler's number, and x is the input value to the function.

One of the key properties of the Sigmoid function is that it saturates at extreme input values, where the output values approach either 0 or 1. The gradients of the Sigmoid function become very small in these regions, leading to the *the vanishing gradient problem* introduced in section 3.2.3. As the gradients diminish, the neural network encounters challenges in efficiently adjusting the weights of the lower layers, leading to a decrease in learning speed and a decline in the overall performance of the model [54].

Tanh

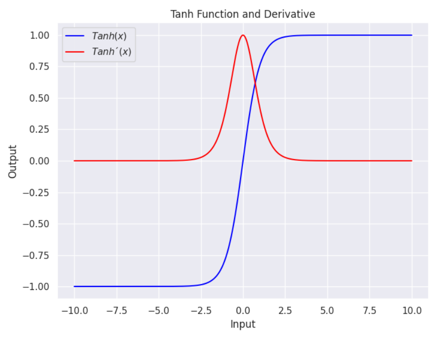


Figure 3.6: The tanh activation function.

The hyperbolic tangent activation function, known as Tanh, is another activation function. It maps any input value to a value between -1 and 1, providing a smooth and continuous transition from the negative minimum value to the positive maximum value [59]. The tanh function is defined mathematically as

$$y = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \tag{3.15}$$

where e is the mathematical constant known as Euler's number, and x is the input value to the function.

The steepness of the Tanh enables it to respond more strongly to changes in the input values than, i.e., the Sigmoid. This property can be beneficial, particularly when training deeper neural networks. However, like the Sigmoid function, the Tanh function can also suffer from the vanishing gradient problem for large positive or negative input values.

ReLU

The Rectified Linear Unit (ReLU) is a simple piecewise linear function that returns the input value if it is positive and 0 otherwise [54].

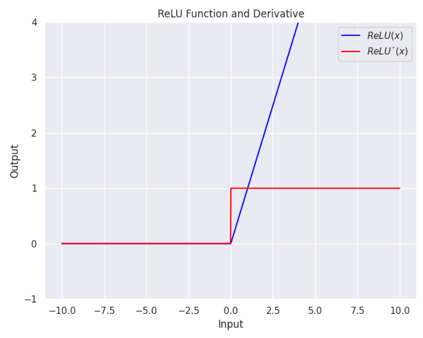


Figure 3.7: The Rectified Linear Unit activation function.

One of the ReLU function's main advantages is its simplicity and computation efficiency. It is computationally cheaper than other activation functions, such as the sigmoid and hyperbolic tangent functions, which involve expensive exponential operations.

$$ReLU(x) = max(0, x) \tag{3.16}$$

Another advantage of the ReLU function is that it can help mitigate the vanishing gradient problem by producing non-zero gradients for positive inputs [54].

A limitation of the ReLU function is that it can suffer from the "dying neuron" problem [60], where some of the neurons in the network may become "dead" and stop producing any output due to a zero gradient. This can happen if the neurons' weights are always initialized to produce negative inputs to the ReLU function, leading to a zero gradient. Several variations of the ReLU function, such as Leaky ReLU and Parametric ReLU, have been proposed to address this issue [54].

3.2.5 Batch normalization

In Section 3.2.3, the problem of vanishing and Exploding gradients is introduced. Batch normalization helps prevent these issues. Normalizing activations through the network prevents small changes to the parameters from amplifying into more significant and suboptimal changes in activations in gradients [61].

When performing Batch Normalization, the output of the hidden layers is normalized using the mean and variance [56] of the batch before or after applying the activation function. This transformation leads to faster and smoother convergence and allows for the utilization of a higher learning rate without compromising the convergence [62].

3.2.6 Skip connections

Skip connections, commonly called residual connections, avoids the vanishing gradient problem by preserving the identity x from the previous layers. It involves the direct connection of the input from one layer to the output of a subsequent layer. This connection allows the information to bypass the intermediate layers and be directly propagated forward [63].

The *Deep residual learning* framework was introduced by He et al. (2015) [64] as a solution to the *degradation problem* which occurs when training deep neural networks. The training gets saturated and degrades rapidly as the depth increases [64]. Figure 3.8 visualize how the identity x is allowed to skip layers through shortcut connections. The output after the skip connection is the element-wise addition of the input that has passed through the residual block, and the identity that has been preserved.

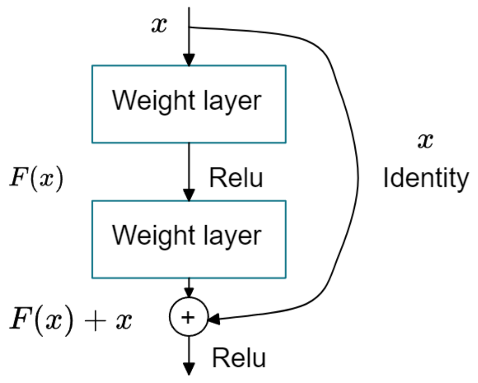


Figure 3.8: A building block of a residual learning framework.

3.3 Learning paradigms in Neural Networks

By leveraging Neural Networks' ability to learn from vast amounts of data, the architectures can process and analyze information, recognize patterns, make predictions, and even generate creative outputs [59]. Its applications range from natural language processing to computer vision.

For different tasks, different optimization techniques are required. These can broadly be categorized into three learning paradigms [7]: supervised, unsupervised, and reinforcement.

Supervised learning happens when the model learns from labeled data. Supervised learning techniques aim to map the input data to the target accuracy, using the labeled training data as ground truth [7]. Examples of supervised learning applications are classification [65] and regression [66].

Reinforcement learning involves an agent that interacts with an environment by taking actions and receiving rewards or penalties in return. The agent aims to develop a policy that maximizes the total cumulative reward over time. The agent operates in discrete time steps, observing the current environment state, selecting actions according to its learned policy, and updating its policy based on the received reward and chosen action to enhance its decision-making skills progressively [7].

Unsupervised learning is a learning-based optimization technique for when there are no labels, mainly used for parameter optimization. The objective is to find regularities in the input data. Real-world data often contains valuable but unexplored information that may not be readily apparent [7]. Unsupervised learning algorithms aim to extract meaningful insights from such data by identifying inherent patterns or groupings.

4 Graph Convolutional Neural Networks

Graphs are used in an increasing number of applications and differ from traditional machine learning algorithms in that the nodes are related to others by links of various types [67]. The route optimization problem of Remiks can be seen as a graph, where the households represent the nodes to be visited, and the roads or distance between them represent the edges.

GNN are deep learning-based methods that operate on graph domain [25]. They can be seen as a generalization of the classical CNN and as an efficient tool to extract complex features in 1D, 2D, and 3D.

4.1 Fundamentals of Graph Neural Networks

GNN use the graph structure and node features X_v to learn a representation vector of a node h_v or the entire graph h_G . State-of-the-art GNNs iteratively update the representation of a node by aggregating representations of its neighbors. After a given number of iterations, the nodes' representation captures the structural information within its network neighborhood [68].

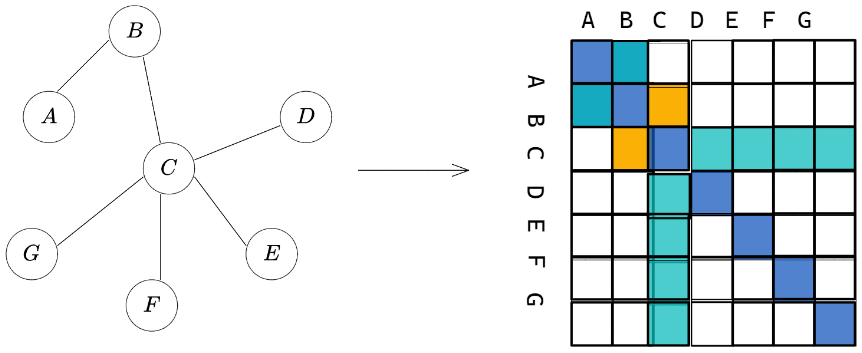


Figure 4.1: Example of how the connectivities in a graph can be visualized in an adjacency matrix. In this scenario, the adjacency matrix is symmetric.

The connectivities of a graph are exemplified as an adjacency matrix in Figure 4.1. The mapping ensures that each node is mapped to its respective representation is either mapped to its respective representation [68].

Figure 4.2 presents four types of information found in graphs that can aid in making predictions: node features, edge features, global context, and connectivity [69]. Node features comprise node identity and the number of neighbors, while edge features encompass edge identity, edge weights, and direction. Global attributes pertain to the number of nodes and information about the global paths. A GNN can be defined as an optimizable transformation on all graph attributes that preserves all graph symmetries [69]. The connectivity concerns which nodes are linked to each other and remain constant during the optimization.

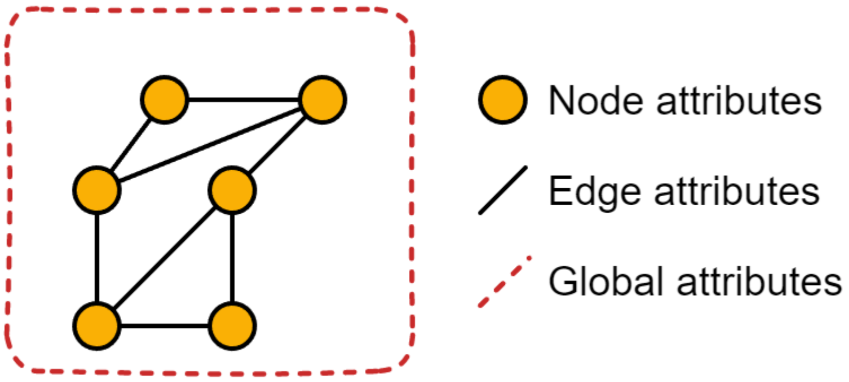


Figure 4.2: Overview over graph attributes: Nodes, edges, and global context. A GNN is an optimizable transformation on all graph attributes that preserves graph symmetry [69].

The nodes can be used to generate a node feature matrix N by assigning each node an index i and storing the feature for $node_i$ in N. The connectivities can be stored as adjacency lists. These describe the connectivity of edge e_k between nodes n_i and n_j as a tuple (i, j) in the k-th entry of the adjacency list [69]. In the Remiks use case, the node feature matrix is generated from the node coordinates, letting the road map represent the connectivities.

4.2 Grapg Neural Network Architecture

The GNN has rapidly emerged as a powerful framework for machine learning on graph-structured data. This section will dive deeper into the fundamental building blocks of a GNN.

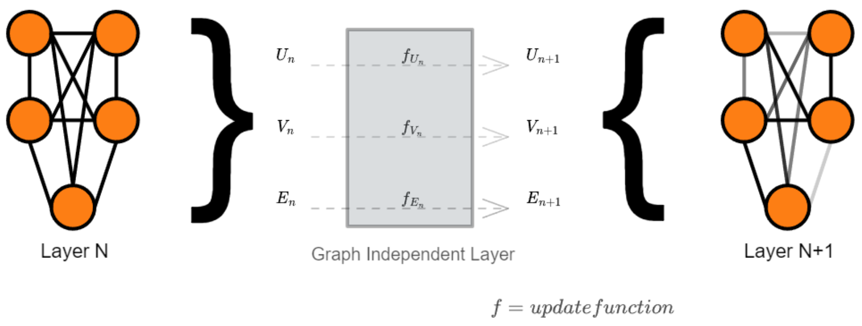


Figure 4.3: A single layer in a GNN, where f represents the update function. Each component, U (global), V (nodes), and E (edges), gets updated to produce a new graph. Each layer n in the GNN represents a separate update function at a different graph attribute. The figure is inspired from [69].

For each layer in a GNN, an update function is applied to the three attributes of the graph. The output of this transformation is a new graph with new graph attributes [69]. Because the connectivity of the input graph is invariant, the output graph of the GNN can be described with the same adjacency list and the same number of feature vectors as the input graph. The embeddings, however, are different.

In order to make predictions on the graph, a final classifier is necessary. Depending on what features are available to use for classification, information can be transferred between attributes. This is performed through pooling operations. The embeddings of the items that are to be pooled are aggregated into a matrix through, i.e., a sum operation [69], and the classification can be performed.

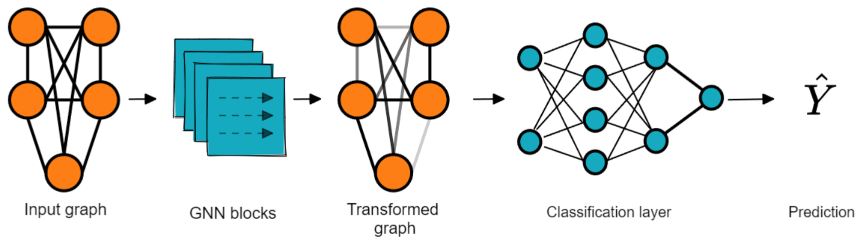


Figure 4.4: A GNN operates by taking an input graph and passing it through a series of layers, each of which updates the attributes of the graph. The output graph is then used to make predictions through a final classifier, which is selected based on the specific optimization goal.

4.3 Graph Convolutional Networks

Message passing and convolution are operations to aggregate and process information of an element's neighbors in order to update the element value [69]. Graph convolution differs from image convolution in the way that neighboring nodes in a graph can be variable, while a pixel has a fixed set of neighboring elements. Graph Convolutional Networks are an extension of CNN from regular grids to irregular graphs [70].

Figure 4.5: Comparison between a convolutional operation for a CNN and a GNN.

By stacking message-passing GNN layers together, one node will eventually capture information from across the entire graph.

Message passing in GNN is a method to make the learned embeddings aware of the graph connectivity, where neighboring nodes or edges exchange information and influence each other updated embeddings [69]. The method can be summarized in three steps [69].

1. For each node in the graph, the algorithm gathers all the embeddings of its neighboring nodes, which are represented as messages.

$$h_v^{(0)} = x_v \tag{4.1}$$

Node v's initial embedding is the original feature vector x_v.
For v's embedding at step k [71]

$$h_v^{(k)} = f^{(k)}\left(\theta^{(k)} \cdot \frac{\sum_{u \in \mathcal{N}(v)} h_u^{(k-1)}}{|\mathcal{N}(v)|} + B^{(k)} \cdot h_v^{(k-1)}\right) \quad \text{for all } v \in V, \tag{4.2}$$

where $\frac{\sum_{u \in \mathcal{N}(v)} h_u^{(k-1)}}{|\mathcal{N}(v)|}$ is the mean of v's neighbors embedding at step $k-1$, $B^{(k)}$ is the decision variables and h_v^{k-1} is node v's embedding at step $k-1$.

2. The messages are aggregated using an aggregate function, such as sum or mean. This step produces a single message for each node that encapsulates information from its neighbors.

$$a_v^{(k)} = AGGREGATE^{(k)}(\{h_{v-1}^{k-1} : v-1 \in \mathcal{N}(v)\}) \tag{4.3}$$

   The neighborhood aggregation of node v in layer k is expressed using the activation of neighboring the neighboring node $v-1$, h_{v-1} of layer $k-1$ [72].

3. The pooled messages are passed through an update function, which typically takes the form of a neural network. Update functions were introduced in Chapter 2.2. This function updates the node embeddings based on the aggregated messages, allowing nodes to incorporate information from their neighbors into their representations.

By repeating this process for multiple layers, a GNN can learn to capture complex relationships and patterns in graph-structured data, making it a powerful tool for various machine-learning tasks.

4.4 Challenges and opportunities in Graph Neural Networks

Graph neural networks (GNNs) have shown great promise in solving complex optimization problems in various domains, including routing problems. In recent years, there has been increasing interest in utilizing machine learning techniques, particularly GNNs, within the Operations Research (OR) field to improve the accuracy and efficiency of optimization problems [73].

One of the most prominent applications of GNNs in OR is the VRP. In this problem, the goal is to find the optimal path or sequence of paths between various nodes in a network. The VRP can be challenging, mainly when the network is large and complex, and there are many possible routes that a vehicle can take.

A GNN can analyze the interactions between the different nodes in a network and learn to predict the optimal path between them, by leveraging the inherent structure of the network [23].

Several recent studies have explored the applicability of GNNs in solving routing problems. For example, Kool et al. (2018) [13] proposed an effective attention-layer-based model across VRP with up to 100 nodes. Similarly, Li et Yan (2021) [74] presented a learning-augmented local search framework aimed at solving large-scale VRP. Arnold et al. (2019) [75] propose an algorithm that utilizes machine learning to optimize the neighborhood ranking to provide a good foundation for route optimization. The results were promising but did not compare against the benchmark models using random initiation with no computational complexity [76].

Overall, GNNs offer exciting opportunities for improving the accuracy and efficiency of VRPs in and other related domains. Their ability to learn from historical data and leverage the inherent structure of networks makes them a promising tool for addressing complex optimization challenges.

5 Green Vehicle Routing Problem

The Green Vehicle Routing (GVRP) field was introduced by Erdogan and Miller-Hooks in 2012 [77] and is rapidly gaining prominence. In their recent article "Green vehicle routing problem: A state-of-the-art review" [12], the authors comprehensively analyzed 323 papers published between 2000 and 2020 on the theme. Notably, 95% of these papers were published during the last decade.

This section introduces several Green Vehicle Routing variants discussed in the literature and highlights important measures to consider when optimizing routes. Furthermore, a framework for calculating fuel consumption in high elevation areas [11] is presented.

Road transportation significantly contributes to increasing atmospheric pollution [48], and the European Commission's key goal for 2030 is to cut at least 40 percent in greenhouse gas emissions compared to the 1990 levels [78]. The instantaneous engine-out emissions rate for Greenhouse Gas (GHG) is directly related to the rate of fuel use [79]. For these reasons, reducing fuel consumption is of great importance to Remiks.

5.1 From VRP to GVRP

Numerous authors [12, 11, 48, 79] have asserted that by expanding the typical VRP to consider not just the economic impact but also the environmental and social impact, there is potential to reduce Carbon Dioxide (CO_2) emissions. Bektas and Laporte (2011) [79] describe the Pollution Routing problem as a case where each vehicle emits an amount of GHG when traveling over a node (i, j). They have found that emissions depend on several factors, such as gravity, slope, load, and speed. The gravity and slope variables are fixed, whereas the load and speed variables can be controlled. Asghari et al. (2020) [12] define the Green VRP as the research of minimizing energy consumption to overcome pollution in transportation activities.

Today, the Remiks fleet consists of conventional fossil fuel-powered vehicles only.

5.2 Classification of Green Vehicle Routing Literature

This section will introduce some directions of GVRP relevant to the Remiks problem that has been proposed in the literature [12].

Type of Engine Different pollution rates and restrictions apply depending on the choice of engine. The investment cost of other engines, charging stations, etc., has to be considered. This work will consider diesel engines only, as Remiks's waste collection vehicles today are diesel-powered. In literature, routing problems for alternative fuel-powered engines often concern electric-powered vehicles or bio-fuel-powered vehicles [80, 81, 82, 83].

Objectives Green VRP is a multi-objective VRP [8]. Sustainability is a business principle encompassing environmental, social, and economic considerations, known as the triple bottom line. A business is sustainable when it achieves equal profitability in all three areas. The triple bottom line is vital for any company, as customer and employee satisfaction is essential for long-term economic profit. In the same way, a business will not last by having a low carbon footprint alone. It is necessary to have financial gain as well.

One way of achieving sustainability and lowering the Remiks vehicles' fuel

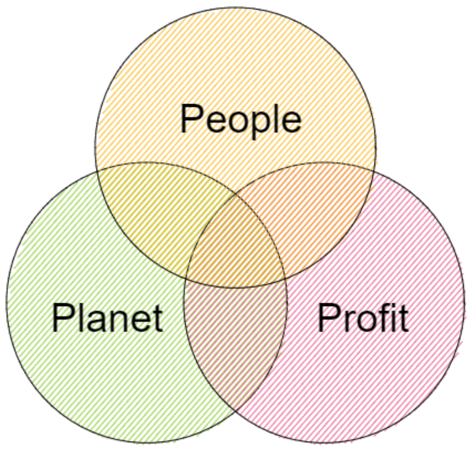


Figure 5.1: Visualization of the Triple Bottom Line [84]. The Triple bottom line is a business concept that measures environmental, social, and economic value.

consumption is to incorporate the aspects of the Triple Bottom Line into the objective function [12]. Economic objectives could be reducing travel or fuel costs, while environmental objectives could be reducing fuel consumption or CO_2-emissions. Social objectives can be customer- or employee satisfaction or minimizing risk. Sustainability can be introduced through the objective function, as one of the multiple objective functions, or be presented as constraints or penalties to the problem [12].

Scenarios While translating a real-world problem into a tractable mathematical formulation, verifying and evaluating the simplification and calculations is essential. Hence, choices must be made regarding the degree of realism, strategic decisions, and restrictions [23].

Examples of decisions regarding the degree of realism are whether to use Euclidean distance or real road distance. For our experiments, real road distance and inclination are included to account for the topography of Tromsø. As the city center of Tromsø is located on an island, Euclidean distance might not produce feasible solutions. Further, several neighborhoods on the Tromsø island are in areas with steep inclinations. Steep roads are generally associated with high fuel consumption [11], and in Tromsø, avoidance of steep arcs is especially important due to icy roads during the winter months. The Map in Figure 5.2 displays the central parts of Tromsø and how the nodes are situated.

Strategic constraints encompass various factors, such as vehicle types, the prescribed number of fuel stations, and the optimal capacity [85]. Additional examples of potential constraints include time windows specifying drivers' working

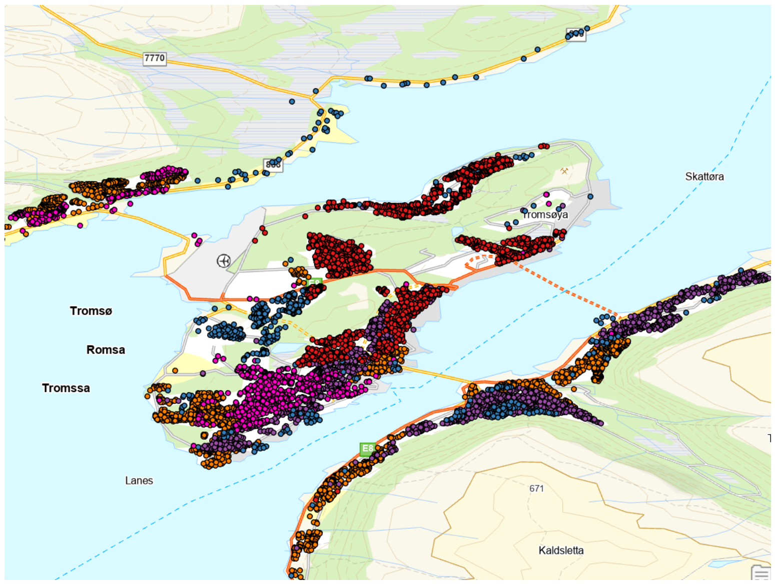


Figure 5.2: Map of the central parts of Tromsø. The circles on the Map visualize households. The color of the circles indicates what day they are visited by the Remiks' vehicles.

hours or break times, total load limitations, and more. In the case of Remiks, this optimization involves the simultaneous utilization of five vehicles considered within the optimization framework. The objective is to optimize Remiks' operations under the current conditions, ensuring practicality and ease of implementation. The scenario which we optimize for will be further described in Chapter 7.

Interaction with traditional VRP The Green VRP is primarily focused on addressing environmental concerns. However, there exists a diverse range of VRP types that can be integrated with Green-VRP models [85]. The selection of a specific VRP type depends on the routing problem's objectives and relevant constraints.

Solution methodologies As described in Chapter 2, routing problems can be solved using various methodologies. Just as traditional VRP, the GVRP can be solved using exact, heuristic, or metaheuristic methods, either individually or combined [86].

5.3 Fuel calculation methods proposed in literature

Green logistics has received increasing and close attention from governments and business organizations in the last decade [87]. This section will broadly present promising methods for modeling fuel consumption in green vehicle routing problems.

Huertas et al. (2022) [11] have proposed a method to accurately determine the fuel consumption of cargo vehicles per kilometer, considering the influence of altitude, road gradient, and vehicle age. The study found that various factors, such as vehicle technology, maintenance levels, fuel characteristics, driver behavior, and external factors like topography, road conditions, and traffic, all play a significant role in determining the actual fuel consumption of the vehicle.

It has been demonstrated that inefficient driving behaviors and traffic congestion can significantly contribute to higher fuel consumption [88, 89]. Therefore, many freight transport companies have installed GPS systems to track real-time position, fuel consumption, and other engine operating variables. Remiks is a company that utilizes this data to improve fuel economy, enhance safety, and optimize vehicle utilization [11].

Woensel et al. (2001) [90] propose a model for measuring the impact of vehicle speed on air pollution. They consider both static factors, such as the vehicle specificities and road infrastructure, and the dynamic traffic flow parameter. Their proposed emission model evaluates the consequences of using constant speed as a parameter for calculating emissions. It concludes that for speeds from $0-37km/h$ and from $105km/h$ and above, emissions are underestimated using constant speeds. They also found that emissions are overestimated in speeds from $37-105km/h$ using constant speeds. The minimum CO emission happened when the vehicle drove at a constant speed of $71km/h$. The Remiks vehicles will have a constant speed below $37km/h$ in urban areas. Woensel et al. propose a method for accurately calculating emissions for vehicle driving in velocities below $37km/h$.

In 2011, Bektas and Laporte [79] introduced the Pollution Routing Problem to reduce overall CO_2 emissions by efficiently dispatching vehicles. Their approach incorporates vehicle distance, travel time, fuel consumption, road angle, load, and GHG emissions. The authors also suggest modeling fuel consumption as a function of speed, similar to Woensel et al. (2001) [90]. However, Bektas and Laporte [79] focus on vehicles with speeds above $40km/h$, as lower-speed vehicles require a different approach. They found that acceleration and deceleration caused by frequent stops significantly contribute to emissions. Bektas and Laporte [79] concluded that minimizing emissions does not necessarily reduce economic cost, as labor cost and the number of vehicles are the dominant economic factors. Using fewer vehicles can also reduce environmental costs. Additionally, they observed that minimizing distance does not necessarily minimize fuel or driver costs. Minimizing financial cost often results in higher energy consumption.

The following chapter will explain the adopted fuel calculation algorithm proposed in this thesis. The algorithm was developed in the review of the previously outlined scientific literature.

Part II

Method and data

6 The Deep Policy Dynamic Programming method for Green Waste Vehicle Routing

Having established a foundation in the theoretical background of operational research, graph neural networks, and green vehicle routing, we are now ready to introduce the Deep Policy Dynamic Programming algorithm [1], and its adaptation to solve Green Vehicle Routing Problems.

The primary objective of this algorithm is to combine the strengths of learned neural heuristics with those of Dynamic Programming to develop sustainable routes for waste collection in cities with varying topography. To achieve this, we will first formulate a framework for calculating fuel consumption based on topography and distance. This framework will be leveraged to create a cost matrix that serves as input data for a pre-trained Graph Convolutional Network. Subsequently, Dynamic Programming will be employed at the output prediction of the network to generate feasible routes optimized for minimizing fuel consumption.

This section introduces a framework that employs the Deep Policy Dynamic

Programming algorithm to tackle Green Vehicle Routing Problems in cities characterized by diverse topography. More details on the data are provided in the next chapter.

6.1 Fuel consumption determination

The energy needed for a vehicle to move is determined by the forces that resist its motion, which include Rolling resistance (F_r), Drag (F_d), gravity F_g, and inertial forces. The power produced by the engine (P_e) is the aggregate of the power required to overcome each of these forces, divided by the mechanical efficiency of the powertrain [11]. The fuel consumption of the vehicle is directly linked to its power output. Additionally, the CO_2 emissions of the vehicle are proportional to the amount of fuel it consumes, as stated by Huertas et al. (2022) [11].

To calculate the vehicle's fuel consumption, it is necessary to consider several factors, such as the vehicle's speed, road conditions, engine efficiency, load capacity, and driving behavior. These factors determine the vehicle's fuel consumption during a tour.

Figure 5.1 shows a simplified figure of the forces that act on a vehicle.

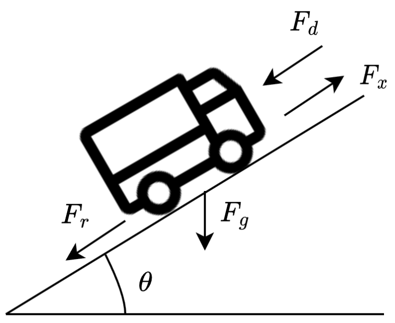


Figure 6.1: Simplification of the forces that act on a vehicle in constant velocity. F_r represent rolling resistance , F_d represent drag forces and F_g represent gravitational forces.

F_r represent rolling resistance , F_d represent drag forces and F_g represent gravitational forces. The right-hand part of Equation 6.1 represents the chemical energy contained in the fuel consumed by the vehicle [11].

Huertas et al (2022) [11] describe the power delivered by the engine as follows,

$$P_e = \frac{(F_d + F_r + F_g + MM_{fi}a)V}{\mu_m} = v_f \rho_f LHV \mu_{th}. \tag{6.1}$$

The forces acting on the vehicle are defined as follows,

$$F_d = \frac{1}{2} c_d \rho_a A_f V^2, \tag{6.2}$$

Where,

V is the velocity,
c_d is the drag coefficient of the vehicle,
A_f is the area of the vehicle,
ρ_a is the density of air.

$$F_r = f_r Mgcos\theta \tag{6.3}$$

and

$$F_g = Mgsin\theta, \tag{6.4}$$

Where,

f_r is the friction coefficient of the road,
M is the total mass of the vehicle,
g is the gravitational force, and
θ is the road gradient in degrees.
μ_{th} is the engine thermal efficiency

$$M_{fi} = 1 + 0.04N_{TDi} + 0.0025N_{TDi}^2, \tag{6.5}$$

Where,

N_{TD} is the engines' transmission ratio.

To determine the route with the least environmental impact, the volumetric flow rate v_f is calculated for all nodes and presented as a distance matrix. The volumetric flow rate is presented in g/s.

$$v_f = (P_e/(LHV \rho_f \mu_{th})) \tag{6.6}$$

Here, ρ_f is the density of the fuel.

Further, the volumetric flow rate is multiplied with a time use matrix in seconds, resulting in a matrix of fuel consumption in grams per distance. This is transformed into liters by multiplying the matrix with the density of diesel in L/g.

The resulting matrix represents the fuel consumption per Liter between all edges in the input data set. Now, we can use this matrix as input in the Graph Convolutional Network.

In order to perform multi-objective optimization, we use the weighted sum method [91], and use α as a scaling factor. We obtain an estimate of the fuel consumption from the distance matrix using the mean fuel efficiency of the Remiks vehicles, μ_{fuel}.

We now have two matrices that provide insights into fuel consumption. The matrix $fuel$ considers the fuel consumption associated with changes in elevation, while the matrix $fuel_{dist}$ takes into account the fuel consumption related to driving distance.

When combining these, the following equation has been used:

$$min \sum_{i=0} (fuel_{dist} + \alpha \cdot fuel)\,, \tag{6.7}$$

Where,

$fuel$ represents the calculated fuel matrix,
$fuel_{dist} = \frac{Distance}{\mu_{fuel}} = \frac{km}{km/L} = L,$
and
$\mu_{fuel} = 1.34 km/L.$

6.2 Graph Convolutional Network for extracting promising edges

Having derived an algorithm for calculating the fuel consumption matrix, we are ready to employ our model. The figure below shows a simplification of the operations that are performed in the network. The stages in the figure are explained below.

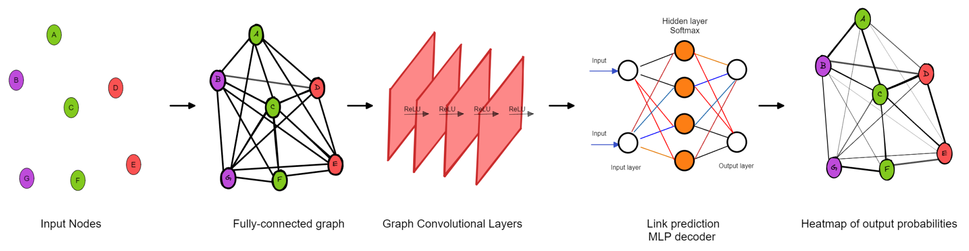


Figure 6.2: Visualization of the Graph Convolutional network, inspired from [15]. The figure shows an example where the nodes are fully connected.

Input layer Initially, two-dimensional coordinates x_i are received as input node features and normalized $x_i \in [0, 1]^2$. These coordinates are embedded into h-dimensional features:

$$\alpha_i = A_1 v_f + b_1, \tag{6.8}$$

where $A_1 \in \mathbb{R}^{h \times 2}$ and $b_1 \in \mathbb{R}^{h \times 1}$ is a bias vector.

Further, the input coordinates are utilized to calculate the fuel consumption v_f using the algorithm described above and embedded as a $\frac{h}{2}$-dimensional feature vector, represented by $d_{i,j}$. The edge input feature β_{ij} is:

$$A_2 d_{i,j} + b_2, \tag{6.9}$$

where $A_2 \in \mathbb{R}^{\frac{h}{2}\times 1}$.

At the input layer $x_i^{l=0} = \alpha_i$ and $e_{i,j}^{l=0} = \beta_{i,j}$. The feature vector x_i^l and edge feature vector $e_{i,j}^l$ refer to the feature vector associated with node i and edge ij at layer l, respectively.

Graph Convolutional Layers The node feature and edge feature at the following layers are defined,

$$x_i^{\ell+1} = x_i^\ell + \mathrm{ReLU}\left(\mathrm{BN}\left(W_1^\ell x_i^\ell + \sum_{j\sim i} \eta_{i,j}^\ell \odot W_2^\ell x_j^\ell\right)\right), \tag{6.10}$$

$$\text{with } \eta_{ij}^\ell = \frac{\sigma\left(e_{ij}^\ell\right)}{\sum_{j'\sim i}\sigma\left(e_{ij'}^\ell\right) + \epsilon},$$

and

$$e_{i,j}^{\ell+1} = e_{i,j}^\ell + \mathrm{ReLU}\left(\mathrm{BN}\left(W_3^\ell e_{i,j}^\ell + W_4^\ell x_i^\ell + W_5^\ell x_j^\ell\right)\right), \tag{6.11}$$

Where $W \in \mathbb{R}^{h\times h}$, is the sigmoid function, ϵ is a small value, ReLU is the rectified linear unit, and BN represents batch normalization.

In arbitrary graphs, diffusion processes are isotropic because there are no specific orientations, making all neighbors equally important. However, this is not always the case, as a neighbor in the same community of nodes may share different information than a neighbor in a separate community. To make the diffusion process anisotropic, we use learnable normalized edge gates $e_{i,j}^l$ that enable pointwise multiplication operations, as proposed by Marcheggiani and Titov (2017) [92].

The batch normalization is performed to enable fast training of deep architectures. As mentioned in Chapter 3, including residual connections is crucial for minimizing the impact of the vanishing gradient issue during backpropagation [20].

MLP classifier and output The edge representations are linked to the ground-truth VRP tour for the training data through a softmax output layer. The model parameters can be trained end-to-end by minimizing the cross-entropy loss via gradient descent.

To calculate the likelihood of an edge being included in the VRP tour of a graph, the edge embedding e^l_{ij} from the final layer is utilized. This probability can be interpreted as generating a probabilistic heatmap across the adjacency matrix of tour links.

Each $p^{VRP}_{i,j} \in [0, 1]^2$ is given by a :

$$p^{VRP}_{i,j} = MLP(e^L_{i,j}) \tag{6.12}$$

Training In many practical scenarios, obtaining or constructing the necessary training data and training the models can be costly or infeasible. To overcome this challenge, instead of engaging in computationally intensive training, we leverage the training data and weights provided by Joshi et al. (2019) for the TSP [20], and Kool et al. (2021) for the VRP [1]. This approach can be seen as a form of One-shot learning [15], wherein we harness the knowledge acquired from one task to enhance learning or performance on a different, yet related, task.

Joshi et al. (2022) propose that training on randomly generated routing problems enables the transfer of the learned policy to larger and more complex real-world scenarios [15].

For the TSP, the model is trained using a training set consisting of 1 million pairs of problem instances and solutions solved with the Concorde solver [93, 20]. For the VRP, the model is trained using 1 million instances, each consisting of 100 nodes, randomly generated based on a fixed distribution [94]. The solutions are obtained using the Lin-Kernighan heuristic [95, 1]. The pre-trained model employs a batch size of 48, a learning rate of 10^{-3}, and undergoes 1500 epochs with 500 training steps. Checkpoints with the lowest validation losses are saved and utilized for testing.

6.3 Deep Policy Dynamic Programming for finding feasible solutions

We can approximate the solution from the derived heatmap using the Deep Policy Dynamic Programming algorithm [1]. Classic dynamic programming was introduced in Chapter 2. The Deep Policy Dynamic Programming is implemented on the sparse heatmap generated for the method described in Section 6.3. The heatmap derives a policy for scoring partial solutions. The Dynamic Programming (DP) algorithm starts with a beam of a single initial solution. Further, it iterates through the following steps:

1. All solutions on the beam are expanded.
2. Dominated solutions are removed from the DP state.
3. The best B solutions, according to the scoring policy, define the beam for the next iteration.

These steps are repeated until the optimal solution is found.

The scoring policy is established using the heatmap values, prioritizing (partial) solutions with the highest total heat while also considering the potential heat for the unvisited nodes. The policy thus selects the B solutions which have the highest score, defined as:

$$score(a) = heat(a) + potential(a), \tag{6.13}$$

where,

$$heat(a) = \sum_{i=1}^{t-1} h_{a_{i-1},a_i}. \tag{6.14}$$

The potential is calculated as follows:

$$potential(a) = potential_0(a) + w_i \sum_{j(a)} \frac{h_{j,i}}{\sum_{k=0}^{n-1} h_{k,i}}, \tag{6.15}$$

Where w_i is the node potential weight given by:

$$w_i = (max(h_{j,i}))(1 - 0.1(\frac{c_{i0}}{max_j(c_{j0})} - 0.5)).$$

The total heat of the edges is determined by adding up the heat values as the solution progresses. When dealing with unvisited nodes, we estimate the potential heat that can be achieved by selecting edges leading to those nodes. This estimation considers the remaining edges available. This approach ensures that important edges are not disregarded in the early stages and remain accessible for future utilization.

By scaling the heatmap values for incoming edges, the remaining potential for node i is initially equal to the maximum possible value w_i. Still, it decreases as better edges become infeasible due to neighboring nodes being visited. The node potential weight w_i is determined by the maximum incoming edge value, which is then adjusted by a factor between 0.95 and 1.05 to give more weight to nodes closer to the start node. This modification encourages the algorithm to retain edges that allow for a return to the start node. The combined heat and potential function efficiently identify promising partial solutions.

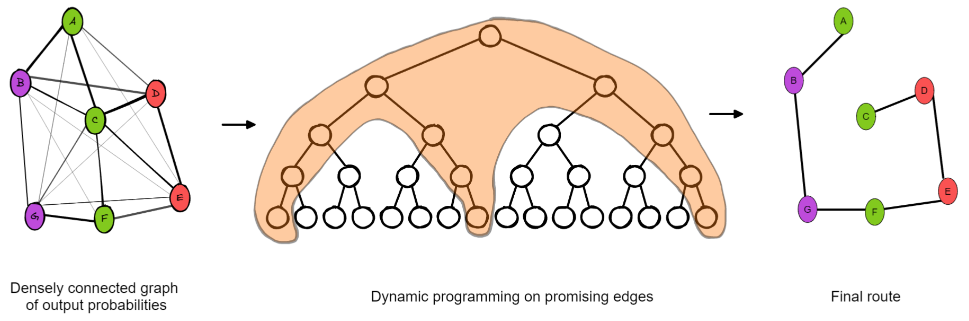

A
B
C
D
E
F
G
Densely connected graph of output probabilities
Dynamic programming on promising edges
Final route

Figure 6.3: Visualization of the Dynamic Programming approach. The Dynamic Programming algorithm is employed on the heatmap of promising edges to limit the algorithm's search space. The Figure is inspired by [1].

The DPDP approach equals traditional Dynamic Programming brute force if using a beam size, $B = n \times 2^n$ for a TSP [1]. In beam search, only a predetermined number of best partial solutions are kept as candidates. Smaller B values allow trading performance for computational cost.

The algorithm keeps track of variables for each partial solution, such as cost, overall distance, current node, and set of visited nodes. Dominated solutions are removed, and the DP state is defined as the tuple of visited nodes and the current node. The algorithm maintains a minimum-cost solution for each unique DP state by removing dominated solutions. The memory required for executing the DP algorithm is O(B), where B is the beam size, allowing for efficient execution in iterations [1].

6.4 Optimization and Hyperparameters

The threshold is a critical parameter for the DPDP approach, as it is used to define the sparse heatmap that the dynamic programming approach is running on [1]. A low threshold rules out most edges, as it will predict close to 0 for all values above the threshold. A higher threshold will rule out fewer edges, resulting in higher computational complexity.

We employ empirical experimentation to identify the optimal threshold for the Remiks dataset. The threshold value is affected by the positioning of the nodes in the data sets and their overall interdependence. When nodes are situated close to each other, a higher threshold is necessary compared to those with more distance between them. It is essential to determine a suitable threshold to avoid the risk of eliminating too many or too few edges. More details on the utilized data are provided in the next chapter.

The experiments below are performed on 1 unit on the GPU cluster operated by the UiT Machine Learning Group [96]. We have performed the experience on nodes within one of Remiks central zones, Friday - zone 2.

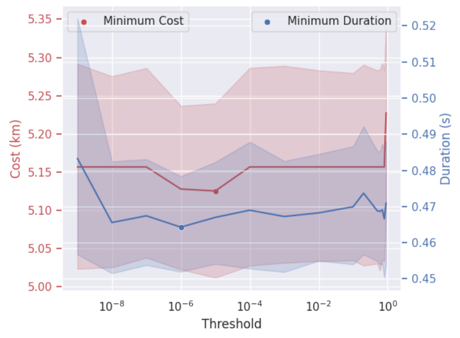


Figure 6.4: Threshold experiment performed to find the optimal tradeoff between computational time and cost.

The experiment displayed in Figure 6.4 was conducted on 20 different instances of 50 nodes for all five zones visited on Thursdays. This involves both district and central zones. We have employed the experiment on threshold values ranging from 10^{-9} to 0.9999. There were no feasible solutions for threshold values above 0.99 for these instances. We observe that a threshold $= 10^{-6}$ uses the shortest computational time, while a threshold $= 10^{-5}$ gives the lowest mean cost for these instances. As our experiments concern a relatively small number of nodes, the computational time is insignificant for these experiments. We, therefore, choose to move forward with threshold=10^{-5}, which yields the lowest mean cost.

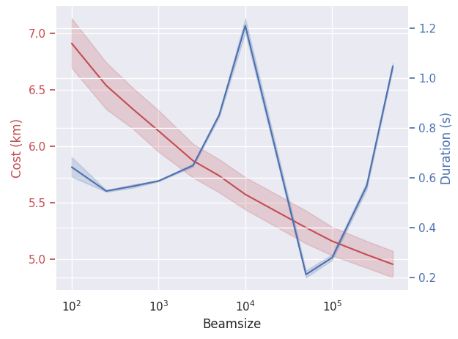


Figure 6.5: Beam size experiments performed to find the optimal tradeoff between computational time and cost.

The experiments displayed in Figure 6.5 is performed on 20 instances of 50 nodes for the same five zones as the Threshold experiments in Figure 6.4. We observe that beam size = 10^5 yields the lowest cost at the minimum time and choose this beam size for future experiments.

6.5 Summary

The strength of the Dynamic Programming Approach is that it is able to solve new, unseen problems in one shot based on policies learned during training [15].

Since the model only requires evaluation once per instance, it can explore numerous beams extensively. The beam size, denoted as B, determines the number of top partial solutions to retain based on the scoring policy.

In accordance with our Problem Definition, we have devised a methodology for calculating fuel consumption using actual distance and elevation matrices obtained via the Google Maps commercial API [21].

We have introduced the Neural Network employed for route optimization and modified the method to minimize fuel consumption, thus fulfilling Target 2 as specified in our Problem Definition.

Furthermore, we have conducted tests to identify effective hyperparameters as starting points for future experiments. We are now prepared to delve deeper into the utilized data set, generously provided by Remiks.

7 Data gathering and augmentations

The effectiveness of machine learning and data analysis tasks depends on the quality and abundance of available data. However, due to various constraints, acquiring precise and diverse datasets can be challenging.

Data gathering encompasses the collection of pertinent data from various sources, while data augmentation techniques strive to enrich existing datasets by introducing variations and augmenting their diversity [23]. This chapter presents the data gathering and augmentation processes employed to create the Remiks data set.

The data regarding the Remiks vehicles and the routes Remiks use today utilized in this project are provided by Remiks [97].

7.1 Understanding the problem

Remiks were briefly introduced in the introduction, Chapter 1. The company has two departments dealing with waste collection: Remiks Household and Remiks Industry [17]. This thesis considers the first one.

In Chapter 5, we introduced some criteria that should be fulfilled in order for a routing problem to be feasible for implementation. In order for the approach to be easily applicable to Remiks, we aim to use as realistic values and constraints as possible. The proposed algorithm optimizes routes explicitly with regard to inclination and real road distance. This is an approach that is not commonly used in literature [11, 98] but which we find promising for creating realistic routes in areas like Tromsø, where distance is not always the most efficient due to mountains and hills.

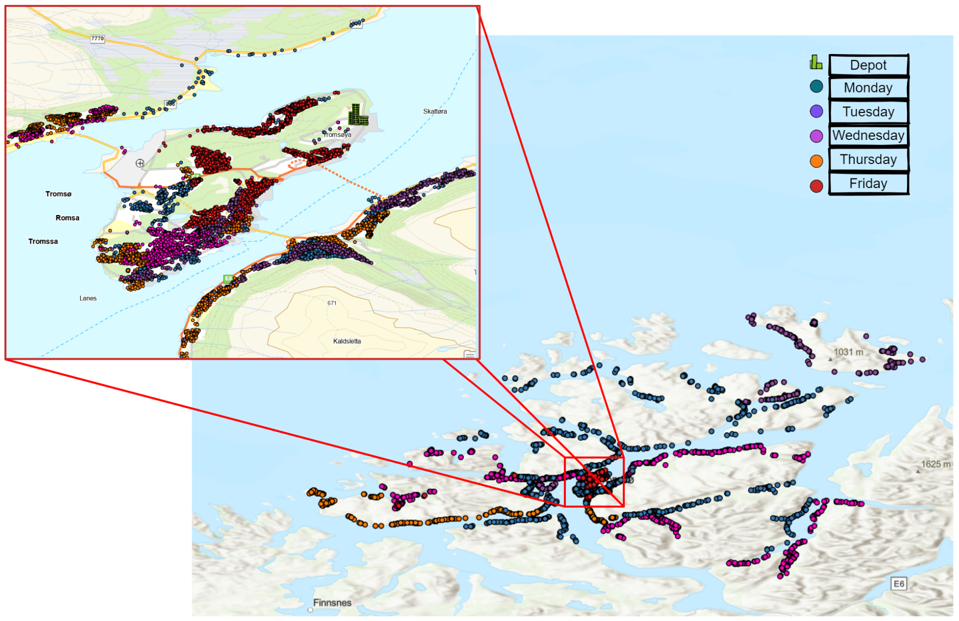


Figure 7.1: Map of all households in Tromsø and Karlsøy municipality, with extra focus on the most central parts of the area. The circles on the map visualize households. The color of the circles indicates what day they are visited by the Remiks vehicles.

Figure 7.1 visualize all nodes where Remiks collect household waste. As the figure indicates, there are many islands, mountain barriers and bays where distance would create infeasible routes. We will look further into this in Chapter 8.

This thesis focuses on five specific Remiks vehicles. Each vehicle is assigned a predetermined set of nodes to collect from on each workday of the week.

Most of the inhabitants in Tromsø and Karlsøy municipality live near the city center, on the city island, which location is displayed in Figure 7.1.

The existing route planning strategy involves assigning a single route to each driver, which they drive repeatedly on a weekly basis and gradually become familiar with. The drivers are given the autonomy to decide their preferred driving approach based on factors such as weather conditions and traffic situations. During winter, when the roads are slippery, drivers often prioritize driving through flat areas of their route initially to allow the vehicle to gain weight and prevent slippage.

One of the main contributions of this thesis is to provide a framework that can facilitate training for the new generation of drivers at Remiks. Currently, the training of new drivers heavily relies on the expertise of experienced drivers. To improve the learning experience for novice drivers, we aim to produce energy-efficient routes that are true to the road map and inclination. We aspire that these routes can serve as valuable tools to aid new drivers in acquiring familiarity with their designated routes.

As a researcher working on an optimization project, it is important to understand the problem to be optimized fully. Meeting the drivers and their leadership group was essential in this process. The conversations with the employees at Remiks were useful in developing an understanding of the problem and how the task is currently being handled.

For the experiment section of this project, we will focus our research on the nodes visited on Fridays, as they are distributed on the city island. On Fridays, four out of the five vehicles are driving routes.

The TSP experiments of Chapter 8 will consider individual zones, while the VRP experiments will consider a randomized selection of the nodes located on the city island in order to explore other zone divisions than the ones currently used by Remiks.

7.2 Data distribution

The GPS system used in Remiks' vehicles [99] collects data on the vehicle's position and speed, expressed in kilometers per hour. These data points are transmitted simultaneously, with an average frequency of one data point every 60 seconds. The telematics system does not report the altitude of the vehicle. To obtain this information, we have relied on the Google Elevation API [21]. The elevation data was extracted as meters above sea level by providing the nodes' location as input.

The Google Distance API [21] has also been utilized to collect the real road distance matrix used to create realistic routes and to plot the obtained routes on the map.

The Vehicle monitoring has been carried out by Co-Driver by Add Secure [99]. The telematics system also reports engine operative variables, including the accumulated fuel consumption. The specific fuel consumption is measured in Liters and is reported for each vehicle at the end of the day.

The input data provided consists of the addresses of the households in Tromsø.

Node distribution in Remiks data compared to training data We introduced the training data used for the DPDP model in Chapter 6.

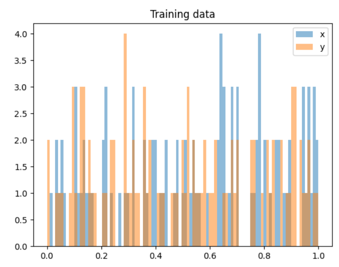
The distribution of nodes in the Remiks dataset deviates from a random distribution. To align the values of the dataset with the scale of the training data used for the model, the x and y coordinates are independently normalized between 0 and 1.

When experimenting with the TSP we have mostly used the zones that Remiks operates within today. Within these zones, the data points are located relatively nearby each other. When normalizing the data points within one zone, the obtained data distribution is pretty similar to the generated data distribution that the model is trained on. The data distribution for TSP and VRP is available for download, provided by Kool et al.(2021) [100] and Joshi et al. (2019) [101].

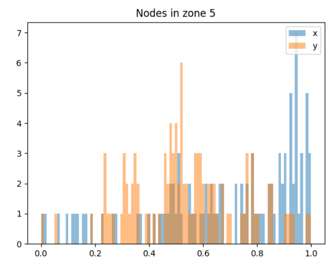
In Figure 7.2, Subfigure a displays the histogram of the randomized training data for the TSP. Subfigure b displays the histogram of the Remiks data of zone 2 on a Friday. We can see that the data points in both data sets are evenly distributed as the zone is located in near proximity to the depot.

Subfigure c and d display the histograms of Zone 1 and 5, with the depot. As visualized in Figure 7.1, the depot of Remiks is located considerably distant from the remaining nodes. Consequently, the nodes are not evenly distributed around the depot. This spatial arrangement poses a unique challenge for optimizing the routing process, as the routes need to account for the significant geographical distance between the depot and the nodes.

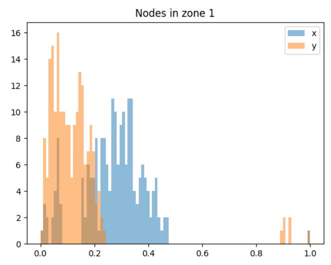
As briefly discussed in Chapter 6, task similarity is essential when utilizing transfer learning. The source task and the target tasks should be related or have some degree of similarity. Therefore, the normalization and preprocessing



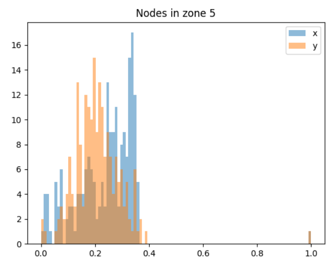
(a) Histogram of the randomized training data.



(b) Histogram of Remiks data for zone 5.



(c) Histogram of Remiks data for zone 1.



(d) Histogram of Remiks data for zone 5.

Figure 7.2: Histograms of data distributions.

of the Remiks coordinates are essential. As the Graph Convolutional Neural network is trained on a large data set of one million instances [100], the model should have learned valuable features and representations that are useful for the Remiks coordinates [58].

7.3 Vehicle specifications

The vehicle specifications and their constraints are important to consider in order to obtain a realistic estimation of fuel consumption.

The Remiks vehicles have a maximum load capacity of 19 tons. Each vehicle weighs 13.5 tons when empty and can carry a maximum of 5 tons of household waste. This limitation is not determined by the weight capability of the vehicle but rather to prevent the degradation of waste due to compression.

Volvo manufactures all vehicles, which are regularly replaced after 6–7 years due to the high maintenance costs and strict uptime requirements. Table 7.1 shows the model specifications of the vehicles studied in this project. As the

Zone	Make	Model	Model year	Fuel type	Wheel layout
1	Volvo	FMX 420	2022	Diesel	4x2
2	Volvo	FMX 420	2022	Diesel	4x2
3	Volvo	FMX 420	2021	Diesel	4x2
4	Volvo	FMX 430	2020	Diesel	4x4
5	Volvo	FMX 430	2020	Diesel	4x4

Table 7.1: Table of vehicle specifications.

vehicle specifications and age does not vary significantly, we will use the same approximation for all vehicles.

7.4 Fuel consumption

As described in Chapter 6, assumptions about a wide range of parameters are required in order to calculate fuel consumption. The choice of these parameters is described below:

Velocity V The Remiks vehicles drive at high velocities on their way to and from pickup zones. In urban areas, vehicles drive at lower speeds and have frequent stops in order to collect waste bins. $3.88m/s$ equals $14km/h$ and is a mean estimation based on the routes that the Remiks vehicles drive in the Tromsø city center.

Drag coefficient c_d The drag coefficient of a vehicle is a dimensionless quantity that represents the resistance the vehicle encounters as it moves through a fluid, typically air. It is a measure of how streamlined or aerodynamic the vehicle is. A lower drag coefficient indicates that the vehicle experiences less resistance and is more efficient in overcoming air resistance. For the experiments described in Chapter 9, $c_d = 0.6$. This estimation is based on a publication by Chowdury et al. (2019) [102].

Frontal area of the vehicle A_f The frontal area of the vehicles is calculated based on information provided by Remiks. The vehicles frontal area of the vehicles is 2.50 meters wide and 4.10 meters high. Thus, the frontal area of the vehicles, $A_f = 10.25m^2$.

Density of the fuel ρ_f The density of fuel refers to its mass per unit volume. The density of diesel is about 0.85kg/L [103]. This is 15-20 percent higher than the density of gasoline.

The coefficient of rolling resistance f_r The coefficient of rolling resistance is a parameter that quantifies the resistance encountered when a vehicle's tires roll over a surface. It measures the energy required to overcome the friction between the tires and the road surface during rolling motion. f_r for a truck tire on an asphalt road should be between 0.006 and 0.01 [104] [105]. In this work, f_r is approximated as 0.008.

Total mass of the vehicle M Remiks have provided numbers regarding the mass of their vehicles. An empty vehicle weighs 13500 kilograms and can carry 5500 kilograms of waste. However, as the household waste gets degraded when compressed, the practical limit is a maximum of 5000 kilograms of household waste at once. The total vehicle mass used for calculations is a static weight of $M = 18500$ kilograms.

Air density ρ_a Air density refers to the mass of air molecules in a given air volume. For this project, we assume constant air density at $\rho_a = 1.293 kgm^{-3}$ [106].

Road gradient θ The road gradient is calculated using elevation obtained and the reasonable road distance in between nodes obtained using Google API [21]. The formula used is

$$\theta = arctan(\frac{\Delta Elevation}{\Delta Road\ distance}) \tag{7.1}$$

Engine thermal efficiency μ_{th} Thermal efficiency is calculated by dividing the useful work output of the engine by the energy input from the fuel. It is expressed as a percentage and represents the portion of the fuel's energy converted into useful work, while the remainder is lost as waste heat [107]. In this work, μ_{th} is estimated to be 0.55 [107].

Mechanical efficiency of the power train μ_m The mechanical efficiency of a powertrain refers to the efficiency with which it converts the input mechanical power into output mechanical power. A diesel engine's lowest theoretical system efficiency is between 55-60 percent [108]. For this project, we use $\mu_m = 0.55$. While engine efficiency focuses on the energy conversion within the engine itself, the mechanical efficiency of the powertrain considers the losses and efficiencies associated with power transmission and delivery to the output device [10].

Low Heating Value LHV Low Heating Value measures the energy released when a fuel is completely burned. The low heating value of diesel is approximately $45.6 kJ/g$ [109, 110].

7.5 Summary

We aim to optimize Remiks' operations while considering realistic assumptions to ensure practicality and ease of implementation. Therefore, the choice of parameters and selection of constraints and objectives are of great importance.

We have collected, analyzed and prepared the data provided by Remiks, and calculated the real distances and elevation matrices using Google Maps Commercial API [21]. This process establish the groundwork for attaining Target 1 and 2 in our problem definition.

Part III

Results and conclusion

8 Results

In this Chapter, we will evaluate the performance of the method introduced in Chapter 6 for two different types of Remiks routes. We will use multi-objective optimization in order to obtain a route that minimizes both fuel consumption and the number of kilometers to drive.

Further, we will evaluate the performance of DPDP for TSP using Euclidean measurements in comparison to the state-of-the-art method Hybrid Genetic Search (HGS) [111] on the Remiks dataset. Lastly, the potential of utilizing the VRP as a means of zone selection is investigated.

For these experiments, we utilized pre-trained models provided by Kool et al. (2021) [1]. We employed pre-trained models on 20 and 50 nodes for the TSP and a pre-trained model of 100 nodes for the VRP.

8.1 Green Vehicle Routing

The implementation of multi-objective optimization enables a thorough analysis of trade-offs involving multiple objectives. This understanding empowers decision-makers to make informed choices that align with their priorities and preferences. In the context of Remiks, it is beneficial to have an objective function that concurrently minimizes both distance and fuel consumption. To establish a suitable relationship between these two factors, a scaling factor known as alpha, α, is employed.

We utilize the equation for the weighted sum method [91], as displayed in Chapter 6:

$$min \sum_{i=0} (fuel_{dist} + \alpha \cdot fuel), \tag{8.1}$$

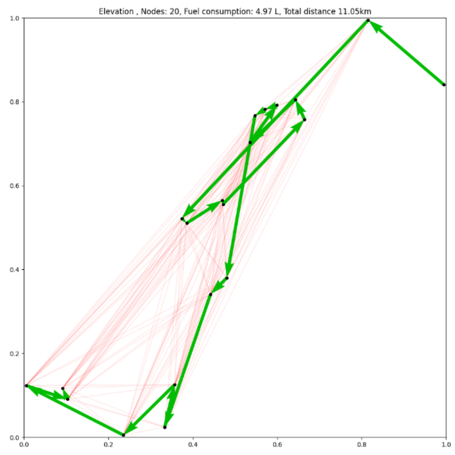
The construction of the Remiks routes is significantly different for routes that visit nodes in the central parts of Tromsø and the routes that cover district areas. To test the robustness of the method, we will test the proposed method on an established Remiks route on the city island and a district zone covering a large area.

8.1.1 Central zone

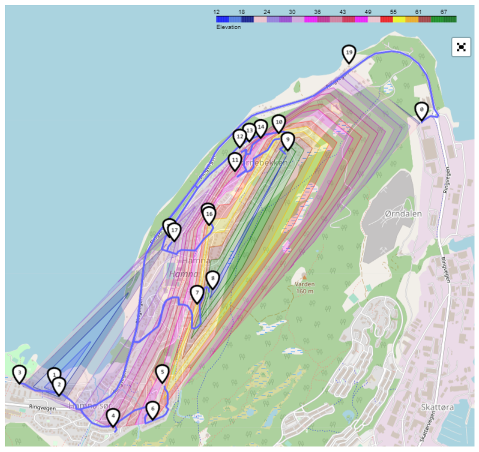
To compare the actual fuel consumption measured by the Remiks vehicle sensors with the estimates used for minimization purposes is challenging. However, the results are valuable when comparing the routes generated by the multi-objective optimization with those calculated using road distance, or elevation only.

We explore an area a few kilometers from the depot. This is the Remiks zone that is most closely connected to Remiks. We chose to run experiments using a sample of 20 nodes to ensure visual clarity and comprehensibility. Figure 8.1 visualize the results obtained by optimizing for fuel consumption including both fuel consumption using Equation 8.1.

Through empirical trials, we found that $\alpha = 0.2$ gave the visually most optimal routes when optimizing for 20 nodes. The experiment is visualized in Figure 8.2. We observe that $\alpha = 0.2$ produces relatively low values for both fuel consumption and kilometers driven. To ensure consistent results, we impose the constraint of having the tour both start and end at the depot.



(a) Fuel minimisation for 20 nodes



(b) Proposed route on map with elevation curves.

Figure 8.1: Experiment with the objective of minimizing fuel consumption based on inclination and driving distance.

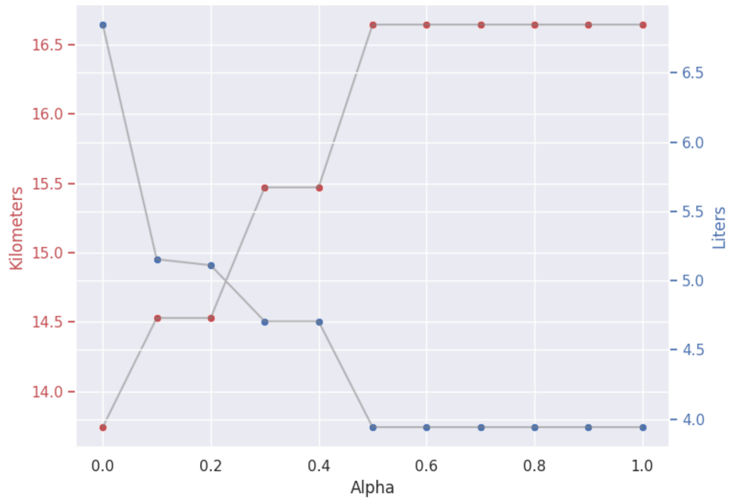
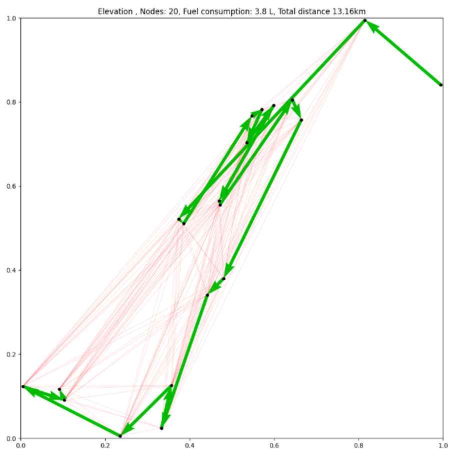
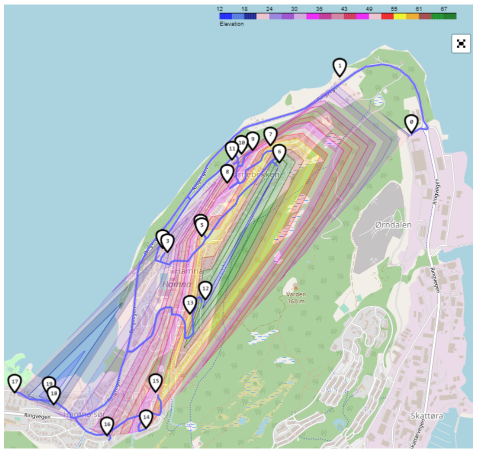


Figure 8.2: Results of experiments performed with different scaling factors, α, for 20 nodes.

Optimization based on inclination If the elevation is the only parameter considered, the model will choose the path with the minimum slope. The design makes the model choose long arcs to make the slope less steep. Further, the model does not account for the additional fuel consumption from the extra kilometers the vehicle travels due to this route choice. This outcome is not ideal, and we need to modify our optimization algorithm accordingly. A route optimized for fuel consumption from inclination only is visualized in Figure 8.3.



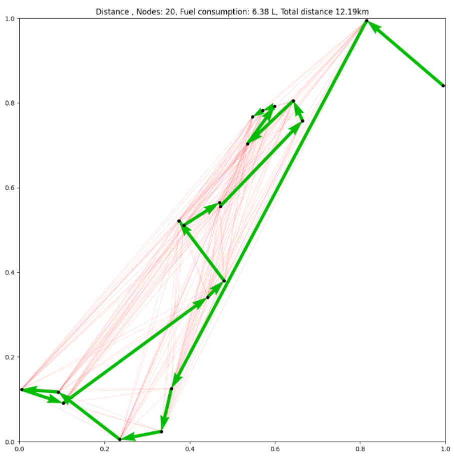
(a) Elevation minimisation for 20 nodes.



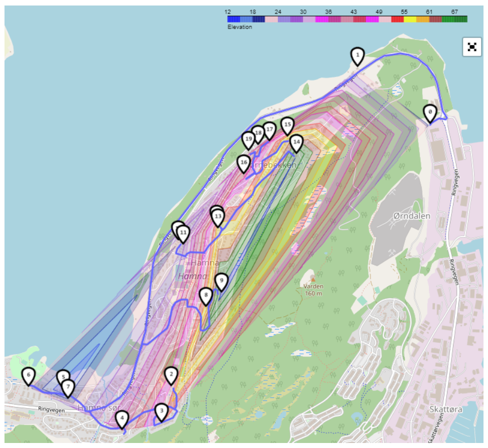
(b) Route on map with elevation curves.

Figure 8.3: Experiment with the objective of minimizing fuel consumption based on inclination.

Optimization based on real road distance When optimizing the routes with road distance as the only parameter being considered, the route visualized in Figure 8.4 is the result. For this area, the proposed route appears like a visually good solution. However, the fuel consumption is higher than the one obtained using the proposed method.

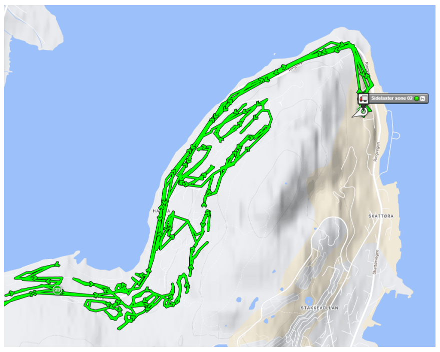


(a) Road distance minimization for 20 nodes.

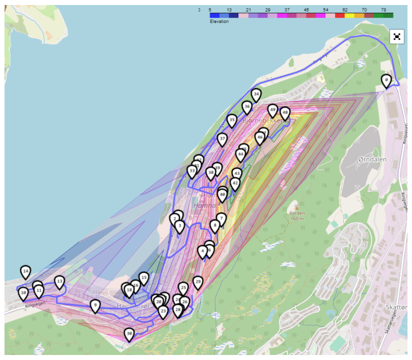


(b) Route on map with elevation curves.

Figure 8.4: Experiment with the objective of minimizing fuel consumption based on road distance.



(a) Remiks route currently used.



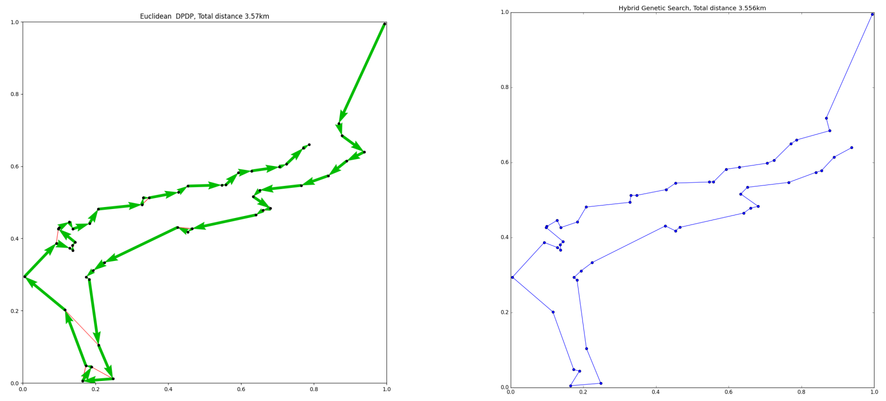
(b) Proposed route on map with elevation curves.

Figure 8.5: Remiks route compared to multiobjective optimization approach.

Comparison against Remiks' current routes Through empirical trials, we found that $\alpha = 0.4$ gave the visually most optimal routes when optimizing for 50 nodes. We compare the route obtained from the proposed method to the route currently employed by Remiks. We observe significant similarities.

Comparison to Hybrid Genetic Search We have compared the results of the DPDP approach on Euclidean distance to the results provided by the HGS [111]. We observe that the two methods yield solutions with many of the same arcs. The total distance traveled is lower for the HGS [111] algorithm, but the difference is insignificant.

Figure 8.5 displays the route generated by DPDP next to the same route generated by the HGS algorithm [111] for the TSP. The HGS algorithm is specifically designed for medium-sized instances of the VRP with up to 1000 nodes [112]. It exhibits notable speed advantages over the DPDP approach for smaller instances. However, when dealing with larger instances, the DPDP shows promise by employing supervised training of a large neural network and requiring only a single model evaluation during test time as it only requires a single model evaluation during test time and is able to generalize well [1].

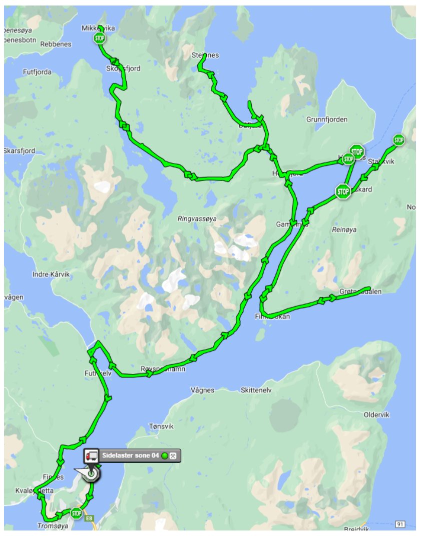


(a) The route optimized with the DPDP [1] approach is 3.57km.

(b) The route optimized with the HGS [111] approach is 3.56km.

Figure 8.6: Euclidean distance route optimization for 50 nodes.

8.1.2 District zone


(a) Current Remiks district route, Monday Zone 4.

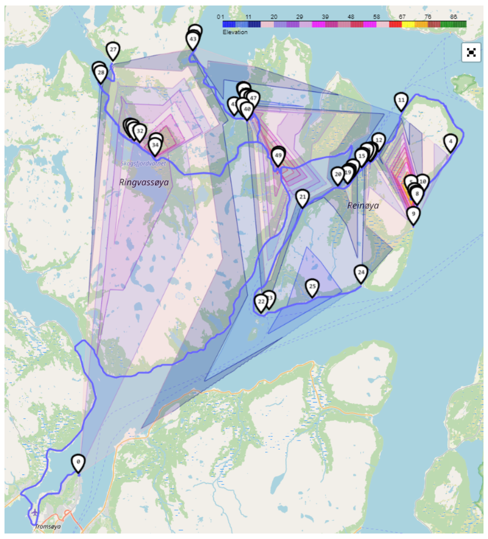

(b) Route optimized to minimize fuel consumption using the weighted sum method [91].

Figure 8.7: Comparison between route currently used by Remiks and route optimized according to the proposed method for minimizing fuel consumption.

Figure 8.7a visualize the route currently used by Remiks for a district zone. Figure 8.7b visualizes the optimized route using the weighted sum method to find a good trade-off between minimum fuel consumption and minimum distance. We observe that the proposed route has significant similarities to the route currently used by Remiks.

Figure 8.8 visualize a plot of the experiment performed to find an optimal value for α.

We observe that the α-value remains the same for all values between 0.4 and 1.0. An $\alpha = 0.2 - 0.3$ gives higher fuel consumption, but fewer kilometers are driven.

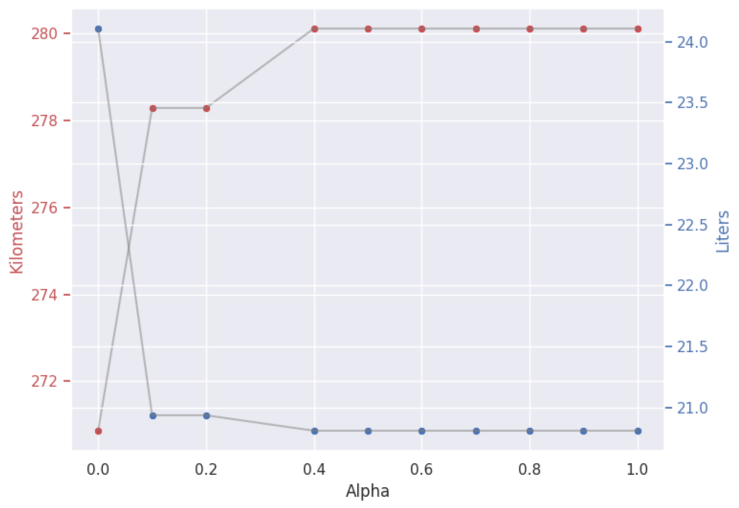
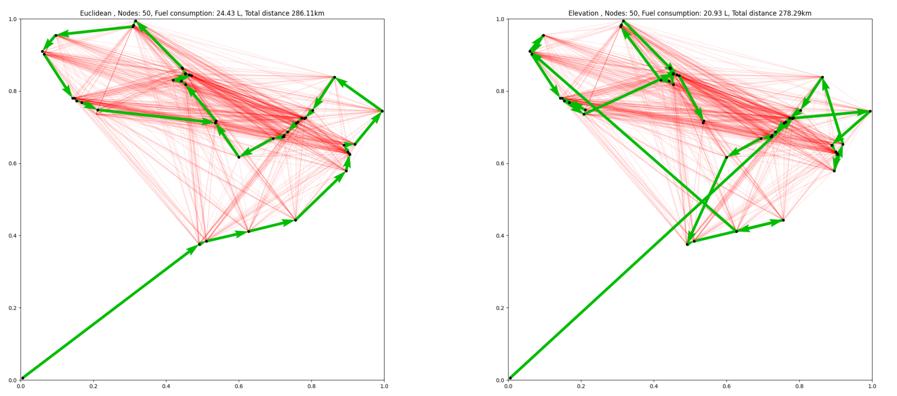


Figure 8.8: Results of experiments performed with different scaling factors, α, for 50 nodes.

In Figure 8.9, it can be observed that optimizing for Euclidean distance leads to higher driving distance and fuel consumption performance compared to the route optimized for fuel minimization. Although the heatmap appears visually more attractive, the resulting outcomes clearly indicate an increase in both driving distance and fuel consumption. A constraint has been introduced to ensure the vehicle concludes its route at the same node in both scenarios to facilitate a fair comparison between the optimizations. This particular node is connected to the highway leading back to the depot.

For the route optimized for Euclidean distance, the Fuel consumption is 24,43 Liters, and the distance traveled is 286,11 Kilometers in road distance. For the route optimized using the proposed method for minimizing fuel consumption, the fuel consumption is 20,93 Liters and the total distance traveled is 278,29 Kilometers.



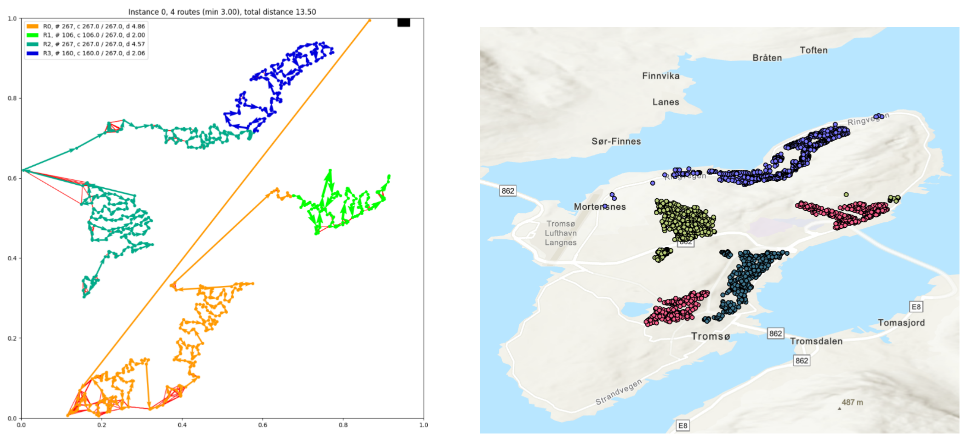
(a) Route optimized with Euclidean distance.

(b) Route optimized for fuel consumption and road distance with $\alpha = 0.2$.

Figure 8.9: Comparison between Euclidian distance optimization and multi-objective optimization on a district route.

8.2 VRP as a means of zone selection

In order to investigate the potential of using vehicle routing as a means for zone selection, we utilize the DPDP framework on the Remiks nodes on the Tromsø city island. Due to limitations with regards to distance and elevation API[21], we minimize the Euclidean distance.


Instance 0, 4 routes (min 3.00), total distance 13.50
R0, # 267, c 267.0 / 267.0, d 4.86
R1, # 106, c 106.0 / 267.0, d 2.00
R2, # 267, c 267.0 / 267.0, d 4.57
R3, # 160, c 160.0 / 267.0, d 2.06
Bråten
Toften
Finnvika
Lanes
Sør-Finnes
Mortensnes
Tromsø Lufthavn Langnes
Tromsø
Tromsdalen
Tomasjord
Strandvegen
487 m

(a) Route optimized with Euclidean distance for 800 nodes.

(b) The Remiks zones currently visited on Fridays.

Figure 8.10: Visualization of proposed zones division for central Tromsø, compared to the Remiks zones.

Figure 1.10a displays a scenario where four vehicles cover 800 nodes. We observe that the routes have similarities to the ones currently used by Remiks.

8.3 Summary

This Chapter has visualized and evaluated the most relevant results obtained in this project. According to Target 3 in our Problem Definition, we have compared the proposed routes to the routes currently used by Remiks. In the next Chapter, we will discuss these results further.

9 Discussion and Future Work

In this section, we assess the effectiveness of the proposed approach compared to Remiks' current methods using metrics introduced in Chapter 2. These metrics include profitability, service quality, equity, consistency, simplicity, and reliability.

9.1 Key findings

9.1.1 Local Route optimization

We applied the proposed algorithm to the data points within the zones currently used by Remiks. Further, we assessed its performance compared to Remiks' current routes and against other optimization techniques.

When it comes to fuel consumption, comparing the actual fuel consumption measured by vehicle sensors with the estimates used for minimization purposes is challenging. The fuel consumption calculation algorithm relies heavily on estimates due to the lack of exact numbers. Moreover, the algorithm currently considers only elevation and road distance variables, while factors such as speed, time usage, and frequency of starts and stops could be beneficial to include. However, the results are valuable when comparing the routes generated by the multi-objective optimization with those calculated using Euclidean distance, road distance, or elevation only. We can make meaningful comparisons regarding which of the three computed routes would have the lowest fuel consumption.

We observed that the routes generated by the proposed algorithm closely resemble the ones currently used by Remiks for central and district zones. Remiks have been using the same type of vehicles for waste collection in Tromsø since 2006 [113], and their employees are well-acquainted with their routes. Moreover, since the drivers finish their day's work when they have collected waste from all households on their routes, it is plausible that their routes are already optimized for efficiency. In Tromsø, where steep hills can become slippery during the winter season, it is advantageous for drivers to avoid driving uphill whenever possible. The drivers can visit households on their route in any order, usually considering weather and driving conditions.

The route optimization generates routes that closely align with Remiks' decisions based on nearly 20 years of experience. This is a significant achievement. The α-parameter can be utilized to explore how to avoid steep roads during icy winter conditions, to select shorter routes during the summer when road conditions are favorable, or as a valuable asset to explore new routes if Remiks changes their infrastructure to electric vehicles.

Further, the proposed algorithm could serve as a valuable tool for training new drivers in selecting the most efficient routes. When new drivers learn their routes, they often have an experienced driver alongside them or an experienced driver available over the phone to guide them. As it is crucial not to miss any areas, new drivers must become familiar with their routes before they begin driving. The ability to experiment with various route options can be a valuable supplementary tool. This could reduce both cost and time for the company.

As Tromsø grows as a city and new households are added, the city's infrastructure will change. The dynamic and easily updatable nature of the route optimization algorithm makes it a valuable tool for determining how to optimize routes in Tromsø in the future, considering factors such as new roads and households.

9.1.2 Global Route Optimization

As the city expands, it becomes necessary to consider extending the number of zones or adjusting their distribution to ensure that the routes can still be completed within working hours on one single day. As the DPDP framework generalizes well from small training instances to large test instances, solving the VRP to determine the optimal selection of zones can be a valuable resource.

This approach, as opposed to traditional clustering methods [114], clusters the nodes based on the actual driving distance between them and how the route can be effectively modeled rather than relying on the nodes' proximity.

From the Results in Figure 8.9, we observe that the algorithm generates routes comparable to the official routes Remiks utilize today. The model is trained on instances of 100 randomly generated points [1] and generalizes well for problems concerning up to 800 nodes.

Further exploration of knowledge transfer from trivial VRP problems to larger problem sizes is a promising area of future research.

9.2 Route attractiveness

Chapter 2 discussed the challenges of finding a suitable measure to evaluate route performance. In this section, we will highlight a few key considerations for the Remiks use case.

Simplicity Rossit et al. (2019) [49] emphasizes the importance of visual attractiveness when optimizing routes. In Chapter 8, Section 1.2, we compared a route optimized for Euclidean distance with one optimized to minimize fuel consumption. We observed that while the former route appeared visually more straightforward and more intuitive, the latter, despite its higher complexity, proved to be more effective in terms of both fuel consumption and distance minimization.

According to Vidal et Laporte (2019) [48], the visual simplicity of a route is significant for creating intuitive courses. However, achieving an optimal route relying on distance is impractical in areas such as Tromsø and Karlsøy municipality, where natural obstacles necessitate routes based on actual elevation and road distance.

Consistency Inhabitants of Tromsø expect their waste to be collected at the same time and day each week. Remiks' current route planning system lacks a fixed order for visiting households, creating challenges in meeting these expectations. Substitute drivers are unaware of the typical collection times for unfamiliar routes. Implementing a system that suggests feasible ways consistently for all drivers would enhance reliability and provide customers with more consistent and predictable collection times.

Reliability Road work, traffic congestion, or accidents can render established routes infeasible. By optimizing routes using the Google Maps API [21], we can access real-time data on travel times for each road segment. This enables us to generate routes that avoid congested or problematic areas. Running the optimization algorithm each morning would provide drivers with updated suggestions to avoid obstacles, enhancing the reliability of the waste collection service.

Addressing these considerations in route optimization can lead to improved route performance, enhanced customer satisfaction, and more efficient waste collection operations. Continued research and development in these areas will contribute to further advancements in waste management systems.

9.3 Can the avoidance of steep hills contribute to lower the fuel consumption of the Remiks vehicles?

Our hypothesis centered on the belief that avoiding steep hills would reduce fuel consumption for Remiks vehicles. Initially, our fuel calculation relied solely on the amount of slope between nodes, considering the real road distance. We discovered that this approach resulted in the optimization algorithm favoring long arcs with low elevation, which was not our intended outcome.

We recognized the need to incorporate a distance measure into the algorithm to address this issue. By considering fuel consumption from elevation and road distance, we can avoid the unintended behavior of prioritizing long, low-elevation arcs. This modification allows us to strike a balance between minimizing the impact of elevation on fuel consumption while still optimizing routes based on overall distance traveled.

Through this adjustment, we aim to create routes that are efficient in terms of fuel consumption and take into account the practical considerations of accurate road distances. We can achieve a more realistic and practical optimization outcome by considering both factors simultaneously.

9.4 Summary

In this Chapter, we have assessed the effectiveness of the DPDP approach in waste management route optimization and outlined several promising lines of research. We have found that the algorithm generates routes that closely resemble Remiks' existing routes, making it a valuable tool for training new drivers. Further research is recommended to refine the optimization techniques and incorporate real-time data and emerging technologies. The importance of route simplicity, consistency, and reliability are emphasized. Our findings indicate that both elevation and distance measures are important factors to take into account when aiming to reduce fuel consumption. Further, we encourage the incorporation of more variable factors such as speed and energy consumed during the start and stop of the vehicles.

According to Target 3 in our Problem Definition, we have evaluated and compared the proposed routes to the routes currently used by Remiks.

10 Concluding remarks

Our research delved into a methodology that facilitates the development of energy-efficient waste collection routes. To augment the existing framework of DPDP [1], we have integrated additional features pertaining to fuel consumption. The implementation and testing of the proposed algorithm have been conducted using real-world waste collection data generously provided by Remiks. Furthermore, the solutions obtained from the algorithm are subject to meticulous evaluation, drawing on Remiks' valuable experience and feedback to assess their efficacy and value. Our findings indicate that both elevation and distance measures are important factors to take into account when aiming to reduce fuel consumption. We propose that the algorithm can be a supplementary tool for enhancing driver training at Remiks.

10.1 Contributions

In this thesis, we have:

1. Integrated a formula for calculating fuel consumption by utilizing obtained distance and elevation matrices, along with relevant data provided by Remiks. We have maintained close and effective communication with Remiks to ensure the proposed algorithm's adaptability to their current system.

2. Successfully extended the DPDP algorithm introduced by Kool et al. (2021) [1] to minimize fuel consumption. By leveraging this framework, we have devised efficient and effective routes considering fuel consumption from both distances traveled and inclinations.

3. Conducted a thorough evaluation and comparison of the proposed routes with the routes currently utilized by Remiks. This evaluation process provides insights into the potential benefits and improvements the proposed routes can offer Remiks regarding operational effectiveness and cost savings.

10.2 Future work

To guide future research, several key directions have been identified:

Scaling up TSP for a Higher Number of Nodes: While the current TSP model is trained to handle 100 nodes, there is a need to expand its capabilities to accommodate a higher number of nodes. Scaling up the TSP model would allow for more extensive route optimization, enabling waste collection services to handle more prominent areas.

Incorporating a more dynamic fuel calculation: To improve the accuracy of fuel consumption estimation, integrating variable factors into the fuel calculation algorithm would be advantageous. A more comprehensive and precise fuel consumption model can be developed by incorporating speed, acceleration, and deceleration variables. Additionally, accounting for the cumulative weight of the vehicles would further enhance the accuracy of the fuel consumption estimation.

Enhancing Slope Measurement Accuracy: The average slope between nodes is used to measure elevation in the fuel calculation algorithm. Future research could focus on developing a more accurate and precise method for measuring slope. This could involve considering factors such as the steepness of individual road segments or the incline/decline profile along a given route. The fuel calculation algorithm can provide more reliable fuel consumption estimates by improving the slope measurement.

Optimize the VRP for Fuel minimization: Currently, the VRP model focuses on optimizing routes based on Euclidean distance. A valuable direction for future work would be integrating fuel consumption minimization in the VRP optimization process. This research has focused on implementing the TSP due to API limitations.

Expanding VRP for a Higher Number of Nodes: The VRP model has shown promising results for up to 850 nodes. However, further advancements are required to extend its capabilities and make it suitable for a more significant number of nodes. By expanding the capacity of the VRP model, achieving more accurate and optimized zone divisions could be possible, improving the overall efficiency of waste collection operations.

Exploring these research areas would contribute to advancing waste management systems, enabling more efficient and optimized route planning for waste collection services. These improvements have the potential to generate cost savings, minimize environmental impact, and enhance customer satisfaction.

Bibliography

[1] Wouter Kool et al. *Deep Policy Dynamic Programming for Vehicle Routing Problems*. 2021. arXiv: 2102.11756 [cs.LG].

[2] Paolo Toth and Daniele Vigo. *The Vehicle Routing Problem*. Ed. by Paolo Toth and Daniele Vigo. Society for Industrial and Applied Mathematics, 2002. DOI: 10.1137/1.9780898718515. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9780898718515. URL: https://epubs.siam.org/doi/abs/10.1137/1.9780898718515.

[3] John E Bell and Patrick R McMullen. "Ant colony optimization techniques for the vehicle routing problem." In: *Advanced engineering informatics* 18.1 (2004), pp. 41–48.

[4] SN Sivanandam et al. "Genetic algorithm optimization problems." In: *Introduction to genetic algorithms* (2008), pp. 165–209.

[5] Bingjie Li et al. "An overview and experimental study of learning-based optimization algorithms for the vehicle routing problem." In: *IEEE/CAA Journal of Automatica Sinica* 9.7 (2022), pp. 1115–1138.

[6] Jean-François Cordeau et al. *New heuristics for the vehicle routing problem*. Springer, 2005.

[7] Ethem Alpaydin. *Introduction to machine learning*. MIT press, 2020.

[8] Ruibin Bai et al. "Analytics and machine learning in vehicle routing research." In: *International Journal of Production Research* 0.0 (2021), pp. 1–27. DOI: 10.1080/00207543.2021.2013566. eprint: https://doi.org/10.1080/00207543.2021.2013566. URL: https://doi.org/10.1080/00207543.2021.2013566.

[9] Michel Deudon et al. *Learning Heuristics for the TSP by Policy Gradient*. URL: https://hanalog.ca/wp-content/uploads/2018/11/cpaior-learning-heuristics-6.pdf.

[10] Oscar Delgado et al. *Fuel eFFiciency Technology in european heavy-DuTy vehicles: Baseline anD poTenTial For The 2020-2030 Time Frame*. 2017. URL: https://theicct.org/wp-content/uploads/2021/06/EU-HDV-Tech-Potential_ICCT-white-paper_14072017_vF.pdf?fbclid=IwAR1BXzxSbCCBF2Nf7-PTKdNAn4Z4oxRGBv2hPVYM9BJ1xk0WwcNcZ_wYnRg.

[11] José I. Huertas et al. "Real vehicle fuel consumption in logistic corridors." In: *Applied Energy* 314 (2022), p. 118921. ISSN: 0306-2619. DOI:

https://doi.org/10.1016/j.apenergy.2022.118921. URL: https://www.sciencedirect.com/science/article/pii/S0306261922003439.

[12] Mohammad Asghari and S. Mohammad J. Mirzapour Al-e-hashem. "Green vehicle routing problem: A state-of-the-art review." In: *International Journal of Production Economics* 231 (2021), p. 107899. ISSN: 0925-5273. DOI: https://doi.org/10.1016/j.ijpe.2020.107899. URL: https://www.sciencedirect.com/science/article/pii/S0925527320302607.

[13] Wouter Kool, Herke van Hoof, and Max Welling. *Attention, Learn to Solve Routing Problems!* 2018. DOI: 10.48550/ARXIV.1803.08475. URL: https://arxiv.org/abs/1803.08475.

[14] Gilbert Laporte. "The traveling salesman problem: An overview of exact and approximate algorithms." In: *European Journal of Operational Research* 59.2 (1992), pp. 231–247.

[15] Chaitanya K. Joshi et al. "Learning TSP Requires Rethinking Generalization." en. In: Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. DOI: 10.4230/LIPICS.CP.2021.33. URL: https://drops.dagstuhl.de/opus/volltexte/2021/15324/.

[16] Wenbin Ouyang et al. *Generalization in Deep RL for TSP Problems via Equivariance and Local Search*. 2021. arXiv: 2110.03595 [cs.LG].

[17] Remiks. *Om Remiks*. URL: https://www.remiks.no/remiks-om-oss/ (visited on 11/16/2022).

[18] Jean-François Cordeau et al. "Vehicle routing." In: *Handbooks in operations research and management science* 14 (2007), pp. 367–428.

[19] Çağrı Koç et al. "Thirty years of heterogeneous vehicle routing." In: *European Journal of Operational Research* 249.1 (2016), pp. 1–21.

[20] Chaitanya K. Joshi, Thomas Laurent, and Xavier Bresson. *An Efficient Graph Convolutional Network Technique for the Travelling Salesman Problem*. 2019. arXiv: 1906.01227 [cs.LG].

[21] 2023. URL: https://developers.google.com/maps.

[22] Astrid Fossum. "Mathematical optimization and the potential of machine learning for the vehicle routing problem within waste collection." In: (2022).

[23] Frederick S. Hillier and Gerald J. Lieberman. *Introduction to Operations Research*. Seventh. New York, NY, USA: McGraw-Hill, 2001.

[24] Sandro Nižetić et al. "Smart technologies for promotion of energy efficiency, utilization of sustainable resources and waste management." In: *Journal of Cleaner Production* 231 (2019), pp. 565–591. ISSN: 0959-6526. DOI: https://doi.org/10.1016/j.jclepro.2019.04.397. URL: https://www.sciencedirect.com/science/article/pii/S0959652619314982.

[25] Jie Zhou et al. "Graph neural networks: A review of methods and applications." In: *AI Open* 1 (2020), pp. 57–81. ISSN: 2666-6510. DOI: https://doi.org/10.1016/j.aiopen.2021.01.001. URL: https://www.sciencedirect.com/science/article/pii/S2666651021000012.

[26] H. Paul Williams. *Model Building in Mathematical Programming*. English. 5th edition. ZSCC: 0002792. Wiley, Jan. 2013.

[27] Gilbert Greefrath and Katrin Vorhölter. "Teaching and Learning Mathematical Modelling: Approaches and Developments from German Speaking Countries." In: *Teaching and Learning Mathematical Modelling: Approaches and Developments from German Speaking Countries*. Cham: Springer International Publishing, 2016, pp. 1–42. ISBN: 978-3-319-45004-9. DOI: 10.1007/978-3-319-45004-9_1. URL: https://doi.org/10.1007/978-3-319-45004-9_1.

[28] Frederick S. Hillier and Gerald J. Lieberman. *Introduction to Operations Research*. Seventh. New York, NY, USA: McGraw-Hill, 2001.

[29] Ron Shamir. "The efficiency of the simplex method: a survey." In: *Management science* 33.3 (1987), pp. 301–334.

[30] P. Narendra and K. Fukunaga. "bnb." In: *IEEE Transactions on Computers* 26.09 (1977), pp. 917–922. ISSN: 1557-9956. DOI: 10.1109/TC.1977.1674939.

[31] Gurobi Optimization. *Mixed-Integer Programming (MIP) - A Primer on the Basics*. URL: https://www.gurobi.com/resources/mixed-integer-programming-mip-a-primer-on-the-basics/ (visited on 12/07/2022).

[32] Richard Bellman. "On the theory of dynamic programming." In: *Proceedings of the national Academy of Sciences* 38.8 (1952), pp. 716–719.

[33] Gilbert Laporte, Paolo Toth, and Daniele Vigo. "Vehicle routing: Historical perspective and recent contributions." In: *EURO Journal on Transportation and Logistics* 2 (May 2013). DOI: 10.1007/s13676-013-0020-6.

[34] G. B. Dantzig and J. H. Ramser. "The Truck Dispatching Problem." In: *Management Science* 6 (Oct. 1959), p. 11. DOI: https://doi.org/10.1287/mnsc.6.1.80.

[35] Gilbert Laporte. "Fifty Years of Vehicle Routing." In: *Transportation Science* 43.4 (2009), 408–416. ISSN: 1526-5447. DOI: 10.1287/trsc.1090.0301. URL: https://doi.org/10.1287/trsc.1090.0301.

[36] Katja Buhrkal, Allan Larsen, and Stefan Ropke. "The Waste Collection Vehicle Routing Problem with Time Windows in a City Logistics Context." In: *Procedia - Social and Behavioral Sciences* 39 (2012). Seventh International Conference on City Logistics which was held on June 7-9,2011, Mallorca, Spain, pp. 241–254. ISSN: 1877-0428. DOI: https://doi.org/10.1016/j.sbspro.2012.03.105. URL: https://www.sciencedirect.com/science/article/pii/S1877042812005721.

[37] Raafat Elshaer and Hadeer Awad. "A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants." In: *COMPUTERS & INDUSTRIAL ENGINEERING* 140 (2020). DOI: 10.1016/j.cie.2019.106242.

[38] Ilhem Boussaïd, Julien Lepagnot, and Patrick Siarry. "A survey on optimization metaheuristics." In: *Information sciences* 237 (2013), pp. 82–117.

[39] Gilbert Laporte. "What You Should Know About the Vehicle Routing Problem." In: *Naval Research Logistics (NRL)* 54 (Dec. 2007), pp. 811 –819. DOI: 10.1002/nav.20261.

[40] Christian Prins. "A simple and effective evolutionary algorithm for the vehicle routing problem." In: *Computers & operations research* 31.12 (2004), pp. 1985–2002.

[41] Yves Rochat and Éric D Taillard. "Probabilistic diversification and intensification in local search for vehicle routing." In: *Journal of heuristics* 1.1 (1995), pp. 147–167.

[42] Michel Gendreau, Alain Hertz, and Gilbert Laporte. "A tabu search heuristic for the vehicle routing problem." In: *Management science* 40.10 (1994), pp. 1276–1290.

[43] Nabil H. Farhat et al. "Optical implementation of the Hopfield model." In: *Appl. Opt.* 24.10 (1985), pp. 1469–1475. DOI: 10.1364/AO.24.001469. URL: https://opg.optica.org/ao/abstract.cfm?URI=ao-24-10-1469.

[44] Irwan Bello et al. "Neural Combinatorial Optimization with Reinforcement Learning." In: *CoRR* abs/1611.09940 (2016). arXiv: 1611.09940. URL: http://arxiv.org/abs/1611.09940.

[45] Michel Deudon et al. "Learning Heuristics for the TSP by Policy Gradient." In: *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Ed. by Willem-Jan van Hoeve. Cham: Springer International Publishing, 2018, pp. 170–181. ISBN: 978-3-319-93031-2.

[46] Raafat Elshaer and Hadeer Awad. "A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants." In: *Computers Industrial Engineering* 140 (2020), p. 106242. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2019.106242. URL: https://www.sciencedirect.com/science/article/pii/S0360835219307119.

[47] Paul A. Sabatier. "Top-Down and Bottom-Up Approaches to Implementation Research: a Critical Analysis and Suggested Synthesis." In: *Journal of Public Policy* 6.1 (1986), 21–48. DOI: 10.1017/S0143814X00003846.

[48] Thibaut Vidal, Gilbert Laporte, and Piotr Matl. "A concise guide to existing and emerging vehicle routing problem variants." In: *European Journal of Operational Research* 286 (Oct. 2019). DOI: 10.1016/j.ejor.2019.10.010.

[49] Diego Gabriel Rossit et al. "Visual attractiveness in routing problems: A review." In: *Computers Operations Research* 103 (2019), pp. 13–34. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2018.10.012. URL: https://www.sciencedirect.com/science/article/pii/S0305054818302715.

[50] C Lee, K Lee, and S Park. “Robust vehicle routing problem with deadlines and travel time/demand uncertainty.” In: *Journal of the Operational Research Society* 63.9 (2012), pp. 1294–1306. DOI: 10.1057/jors.2011.136. eprint: https://doi.org/10.1057/jors.2011.136. URL: https://doi.org/10.1057/jors.2011.136.

[51] Alan Osorio-Mora et al. “The Multi-Depot Cumulative Vehicle Routing Problem With Mandatory Visit Times and Minimum Delayed Latency.” In: *IEEE Access* 9 (2021), pp. 27210–27225. DOI: 10.1109/ACCESS.2021.3058242.

[52] Bingjie Li et al. “An Overview and Experimental Study of Learning-Based Optimization Algorithms for the Vehicle Routing Problem.” In: *IEEE/CAA Journal of Automatica Sinica* 9.7 (2022), pp. 1115–1138. DOI: 10.1109/JAS.2022.105677.

[53] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition, Fourth Edition*. 4th. USA: Academic Press, Inc., 2008. ISBN: 1597492728.

[54] Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2014. ISBN: 0262028182.

[55] 2023. URL: https://www.ibm.com/topics/gradient-descent.

[56] 2016. URL: https://www.universitetsforlaget.no/kalkulus-4-utgave-1.

[57] Michael A. Nielsen. *Neural Networks and Deep Learning*. misc. 2018. URL: http://neuralnetworksanddeeplearning.com/.

[58] Srikanth Tammina. “Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images.” In: vol. 9. Oct. 2019, p9420. DOI: 10.29322/IJSRP.9.10.2019.p9420.

[59] Anne håkansson and Ronald Hartung. *Artificial Intelligence - Concepts, Areas, Techniques and Applications*. Oct. 2020. ISBN: 9789144125992.

[60] Lu Lu. “Dying ReLU and Initialization: Theory and Numerical Examples.” In: *Communications in Computational Physics* 28.5 (2020), pp. 1671–1706. DOI: 10.4208/cicp.oa-2020-0165. URL: https://doi.org/10.4208%2Fcicp.oa-2020-0165.

[61] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG].

[62] Johann Huber. *Batch normalization in 3 levels of understanding - Towards Data Science*. 2020. URL: https://towardsdatascience.com/batch-normalization-in-3-levels-of-understanding-14c2da90a338.

[63] Vighnesh Uday Tamse. *Deep Residual Learning for Image Recognition (ResNet)*. 2020. URL: https://medium.com/analytics-vidhya/deep-residual-learning-for-image-recognition-resnet-94a9c71334c9.

[64] Kaiming He et al. “Deep Residual Learning for Image Recognition.” In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[65] Jose Caceres-Cruz et al. “Rich Vehicle Routing Problem: Survey.” In: *ACM Comput. Surv.* 47.2 (2014). ISSN: 0360-0300. DOI: 10.1145/2666003. URL: https://doi.org/10.1145/2666003.

[66] Donald F Specht et al. “A general regression neural network.” In: *IEEE transactions on neural networks* 2.6 (1991), pp. 568–576.

[67] Zonghan Wu et al. “A Comprehensive Survey on Graph Neural Networks.” In: *IEEE Transactions on Neural Networks and Learning Systems* 32.1 (2021), pp. 4–24. DOI: 10.1109/TNNLS.2020.2978386.

[68] Keyulu Xu et al. *How Powerful are Graph Neural Networks?* 2018. DOI: 10.48550/ARXIV.1810.00826. URL: https://arxiv.org/abs/1810.00826.

[69] Benjamin Sanchez-Lengeling et al. “A Gentle Introduction to Graph Neural Networks.” In: *Distill* (2021). https://distill.pub/2021/gnn-intro. DOI: 10.23915/distill.00033.

[70] Na Hu et al. “Graph learning-based spatial-temporal graph convolutional neural networks for traffic forecasting.” In: *Connection Science* 34.1 (2022), pp. 429–448. DOI: 10.1080/09540091.2021.2006607. eprint: https://doi.org/10.1080/09540091.2021.2006607. URL: https://doi.org/10.1080/09540091.2021.2006607.

[71] Ameya Daigavane, Balaraman Ravindran, and Gaurav Aggarwal. “Understanding Convolutions on Graphs.” In: 6.8 (2021). DOI: https://doi.org/10.23915/distill.00032. URL: https://distill.pub/2021/understanding-gnns/.

[72] Anuradha Wickramarachchi. *What Can You Do With GNNs - Towards Data Science*. 2021. URL: https://towardsdatascience.com/what-can-you-do-with-gnns-5dbec638b525.

[73] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. “Machine learning for combinatorial optimization: A methodological tour d’horizon.” In: *European Journal of Operational Research* 290.2 (2021), pp. 405–421. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2020.07.063. URL: https://www.sciencedirect.com/science/article/pii/S0377221720306895.

[74] Sirui Li, Zhongxia Yan, and Cathy Wu. “Learning to delegate for large-scale vehicle routing.” In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 26198–26211.

[75] Florian Arnold et al. *PILS: Exploring high-order neighborhoods by pattern mining and injection*. Dec. 2019.

[76] Association for Constraint Programmingn. *Integrating Machine Learning into state of the art Vehicle Routing Heuristics - Daniele Vigo*. Youtube, 2021. URL: https://www.youtube.com/watch?v=YxuNJ_4tt3Q.

[77] Sevgi Erdoğan and Elise Miller-Hooks. “A Green Vehicle Routing Problem.” In: *Transportation Research Part E: Logistics and Transportation Review* 48.1 (2012). Select Papers from the 19th International Symposium on Transportation and Traffic Theory, pp. 100–114. ISSN: 1366-5545.

DOI: https://doi.org/10.1016/j.tre.2011.08.001. URL: https://www.sciencedirect.com/science/article/pii/S1366554511001062.

[78] International Energy Agency. *2030 climate energy framework*. 2014. URL: https://climate.ec.europa.eu/eu-action/climate-strategies-targets/2030-climate-energy-framework_en (visited on 11/06/2022).

[79] Tolga Bektaş and Gilbert Laporte. "The Pollution-Routing Problem." In: *Transportation Research Part B: Methodological* 45.8 (2011). Supply chain disruption and risk management, pp. 1232–1250. ISSN: 0191-2615. DOI: https://doi.org/10.1016/j.trb.2011.02.004. URL: https://www.sciencedirect.com/science/article/pii/S019126151100018X.

[80] Shuai Zhang et al. "Electric vehicle routing problem with recharging stations for minimizing energy consumption." In: *International Journal of Production Economics* 203 (2018), pp. 404–413. ISSN: 0925-5273. DOI: https://doi.org/10.1016/j.ijpe.2018.07.016. URL: https://www.sciencedirect.com/science/article/pii/S0925527318302810.

[81] Maurizio Bruglieri et al. "A path-based solution approach for the green vehicle routing problem." In: *Computers & Operations Research* 103 (2019), pp. 109–122.

[82] Shuai Zhang et al. "Ant colony algorithm for routing alternate fuel vehicles in multi-depot vehicle routing problem." In: *Decision Science in Action: Theory and Applications of Modern Decision Analytic Optimisation* (2019), pp. 251–260.

[83] Juho Andelmin and Enrico Bartolini. "An exact algorithm for the green vehicle routing problem." In: *Transportation Science* 51.4 (2017), pp. 1288–1303.

[84] The complexity project. *Complexity and the Triple Bottom Line*. 2020. URL: https://thecomplexityproject.com/complexity-and-the-triple-bottom-line/ (visited on 11/16/2022).

[85] Mohammad Asghari and S. Mohammad J. Mirzapour Al-e-hashem. "Green vehicle routing problem: A state-of-the-art review." In: *International Journal of Production Economics* 231 (2021), p. 107899. ISSN: 0925-5273. DOI: https://doi.org/10.1016/j.ijpe.2020.107899. URL: https://www.sciencedirect.com/science/article/pii/S0925527320302607.

[86] Gilbert Laporte. "The vehicle routing problem: An overview of exact and approximate algorithms." In: *European Journal of Operational Research* 59.3 (1992), pp. 345–358. ISSN: 0377-2217. DOI: https://doi.org/10.1016/0377-2217(92)90192-C. URL: https://www.sciencedirect.com/science/article/pii/037722179290192C.

[87] Canhong Lin et al. "Survey of Green Vehicle Routing Problem: Past and future trends." In: *Expert Systems with Applications* 41.4, Part 1 (2014), pp. 1118–1138. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2013.07.107. URL: https://www.sciencedirect.com/science/article/pii/S095741741300609X.

[88] Hans Jakob Walnum and Morten Simonsen. “Does driving behavior matter? An analysis of fuel consumption data from heavy-duty trucks.” In: *Transportation Research Part D: Transport and Environment* 36 (2015), pp. 107–120. ISSN: 1361-9209. DOI: https://doi.org/10.1016/j.trd.2015.02.016. URL: https://www.sciencedirect.com/science/article/pii/S1361920915000255.

[89] Peng Ping et al. “Impact of Driver Behavior on Fuel Consumption: Classification, Evaluation and Prediction Using Machine Learning.” In: *IEEE Access* 7 (2019), pp. 78515–78532. DOI: 10.1109/ACCESS.2019.2920489.

[90] Tom Van Woensel, Ruth Creten, and Nico Vandaele. “Managing the Environmental Externalities of Traffic Logistics: The Issue of Emissions.” In: *Production and Operations Management* 10 (Nov. 2000). DOI: 10.1111/j.1937-5956.2001.tb00079.x.

[91] R. Timothy Marler and Jasbir S Arora. “The weighted sum method for multi-objective optimization: new insights.” In: 41.6 (2010), 853–862. DOI: https://doi.org/10.1007/s00158-009-0460-7. URL: https://link.springer.com/article/10.1007/s00158-009-0460-7#Abs1.

[92] Diego Marcheggiani and Ivan Titov. “Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling.” In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 1506–1515. DOI: 10.18653/v1/D17-1159. URL: https://aclanthology.org/D17-1159.

[93] Túlio Azevedo and Marco Carvalho. *Concorde solver installation and use*. Apr. 2018. DOI: 10.13140/RG.2.2.17141.27364.

[94] MohammadReza Nazari et al. “Deep Reinforcement Learning for Solving the Vehicle Routing Problem.” In: *CoRR* abs/1802.04240 (2018). arXiv: 1802.04240. URL: http://arxiv.org/abs/1802.04240.

[95] Keld Helsgaun. “An effective implementation of the Lin–Kernighan traveling salesman heuristic.” In: *European Journal of Operational Research* 126.1 (2000), pp. 106–130. ISSN: 0377-2217. DOI: https://doi.org/10.1016/S0377-2217(99)00284-2. URL: https://www.sciencedirect.com/science/article/pii/S0377221799002842.

[96] 2023. URL: https://uitml.github.io/springfield/.

[97] Petter Hofstad Strand Project Leader. *Personal Communication Remiks employees*. 2023.

[98] Kris Braekers, Katrien Ramaekers, and Inneke Van Nieuwenhuyse. “The vehicle routing problem: State of the art classification and review.” In: *Computers Industrial Engineering* 99 (2016), pp. 300–313. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2015.12.007. URL: https://www.sciencedirect.com/science/article/pii/S0360835215004775.

[99] 2020. URL: https://www.addsecure.no/product/co-driver-app/.

[100] wouterkool. *wouterkool/dpdp: DPDP*. 2023. URL: https://github.com/wouterkool/dpdp.

[101] chaitjo. *chaitjo/graph-convnet-tsp: Code for the paper "An Efficient Graph Convolutional Network Technique for the Travelling Salesman Problem" (INFORMS Annual Meeting Session 2019)*. 2021. URL: https://github.com/chaitjo/graph-convnet-tsp/tree/master.

[102] Harun Chowdhury et al. "An experimental study on of the effect of various deflectors used for light trucks in Indian subcontinent." In: *Energy Procedia* 160 (Feb. 2019), pp. 34–39. DOI: 10.1016/j.egypro.2019.02.115.

[103] Mark J. Kaiser and E.W. McAllister. "22 - Specifications." In: *Pipeline Rules of Thumb Handbook (Ninth Edition)*. Ed. by Mark J. Kaiser and E.W. McAllister. Ninth Edition. Gulf Professional Publishing, 2023, pp. 1089–1146. ISBN: 978-0-12-822788-6. DOI: https://doi.org/10.1016/B978-0-12-822788-6.00019-0. URL: https://www.sciencedirect.com/science/article/pii/B9780128227886000190.

[104] 2023. URL: https://www.school-for-champions.com/science/friction_rolling_coefficient.htm#.ZGZIu3ZBw7E.

[105] In: ().

[106] Science Data. *Air Mass/Density*. 2023. URL: https://www.earthdata.nasa.gov/topics/atmosphere/atmospheric-pressure/air-mass-density.

[107] Haoye Liu et al. "Improvement of emission characteristics and thermal efficiency in diesel engines by fueling gasoline/diesel/PODEn blends." In: *Energy* 97 (2016), pp. 105–112.

[108] Volvo Construction Equipment. *Fuel use - how low can you go?* 2018. URL: https://www.volvoce.com/global/en/news-and-events/news-and-stories/2018/fuel-use-how-low-can-you-go/.

[109] 2023. URL: https://www.engineeringtoolbox.com/fuels-higher-calorific-values-d_169.html.

[110] 2016. URL: https://world-nuclear.org/information-library/facts-and-figures/heat-values-of-various-fuels.aspx.

[111] Thibaut Vidal. "Hybrid Genetic Search for the CVRP: Open-Source Implementation and SWAP* Neighborhood." In: *CoRR* abs/2012.10384 (2020). arXiv: 2012.10384. URL: https://arxiv.org/abs/2012.10384.

[112] vidalt. *vidalt/HGS-CVRP: Modern implementation of the hybrid genetic search (HGS) algorithm specialized to the capacitated vehicle routing problem (CVRP). This code also includes an additional neighborhood called SWAP**. 2023. URL: https://github.com/vidalt/HGS-CVRP.

[113] 2018. URL: https://www.remiks.no/remiks-om-oss/var-historie/.

[114] Thibaut Vidal et al. "Hybrid metaheuristics for the Clustered Vehicle Routing Problem." In: *Computers Operations Research* 58 (2015), pp. 87–99. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2014.10.

019. URL: https://www.sciencedirect.com/science/article/pii/S0305054814002767.