UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Obed Kobina Nsiah

Optimized Snowplow Routing on Estonian Roads: Machine Learning Approaches

Master's Thesis (30 ECTS)

Supervisors:
Kallol Roy, PhD
Jaan Übi, PhD

Tartu 2025

Optimized Snowplow Routing on Estonian Roads: Machine Learning Approaches

Abstract:

Efficient management of snowplowing operations on roads is a critical infrastructure challenge in regions with harsh winter conditions, such as Estonia. This thesis addresses the Snowplow Arc Routing Problem (SARP), with the intention of optimizing snowplow routes to minimize deadheading, ensure complete task coverage, and balance workloads between multiple vehicles. The study investigates four primary aspects: (1) the effectiveness of Mixed Integer Linear Programming (MILP) in optimally solving SARP instances; (2) the capability of supervised machine learning approaches to approximate MILP solutions while maintaining connectivity and complete coverage; (3) the potential of reinforcement learning (RL) techniques to improve routing performance and quality within machine learning frameworks; and (4) the comparative advantages and limitations of classical MILP, supervised ML, RL, and hybrid ML-RL models for such tasks. An MILP model implemented with Gurobi provides optimal benchmarks for subsequent learning-based approaches. Although the MILP achieves an optimal deadhead ratio of 0.002, it demonstrates significant computational scalability constraints. A supervised learning model utilizing graph neural networks (GNN) and Transformer-based pointer networks is developed, achieving an average deadhead ratio of 1.47 with initial coverage feasibility of 65.6% before heuristic post-processing. To address supervised learning limitations, an actor-critic RL model is implemented, which shows an improved deadhead ratio of 1.42. A proposed two-stage hybrid ML-RL architecture that combines supervised edge-to-truck assignments with RL-driven routing achieves a better workload balance coefficient of variation of 0.843 compared to 0.903 for supervised and 0.945 for RL, and a better initial coverage feasibility of 70.3%, but a slightly higher deadhead ratio of 1.53 highlighting trade-offs between operational efficiency and workload distribution. These approaches are validated through experiments conducted on real-world road networks in Tartu, Estonia. The results indicate that pure RL provides the most efficient solutions regarding route length and deadheading, while the hybrid ML-RL model demonstrated strengths in workload equity and coverage feasibility.

Keywords: Arc Routing Problem (ARP), Snowplow Routing, Mixed Integer Linear Programming (MILP), Graph Neural Networks (GNN), Reinforcement Learning (RL), Hybrid ML-RL Model

CERCS: P170 - Computer science, numerical analysis, systems, control; P176 - Artificial intelligence

Optimeeritud lumelükkamise marsruutide planeerimine Eesti teedel: masinõppe lähenemisviisid

Lühikokkuvõte:

Lumekoristustööde tõhus haldamine teedel on kriitilise tähtsusega infrastruktuuriprobleem karmide talvetingimustega piirkondades, näiteks Eestis. See väitekiri käsitleb lumekoristuskaare marsruutimise probleemi (SARP), eesmärgiga optimeerida lumekoristusmarsruute, et minimeerida tühimike arvu, tagada ülesannete täielik katvus ja tasakaalustada töökoormust mitme sõiduki vahel.

Uuringus uuritakse nelja peamist aspekti: (1) segatud täisarvulise lineaarprogrammeerimise (MILP) efektiivsust SARP-i eksemplaride optimaalsel lahendamisel; (2) juhendatud masinõppe lähenemisviiside võimet MILP-lahendusi lähendada, säilitades samal ajal ühenduvuse ja täieliku katvuse; (3) tugevdusõppe (RL) tehnikate potentsiaali marsruutide toimivuse ja kvaliteedi parandamiseks masinõppe raamistikes; ja (4) klassikalise MILP-i, juhendatud ML-i, RL-i ja hübriidsete ML-RL-mudelite võrdlevad eelised ja piirangud selliste ülesannete puhul. Gurobi abil rakendatud MILP-mudel pakub optimaalseid võrdlusaluseid järgnevatele õppepõhistele lähenemisviisidele. Kuigi MILP saavutab optimaalse tühimike arvu 0,002, näitab see olulisi arvutuslikke skaleeritavuse piiranguid. Arendati juhendatud õppe mudelit, mis kasutab graafilisi närvivõrke (GNN) ja transformaatorpõhiseid pointervõrke, saavutades keskmise tühimike suhte 1,47 ja esialgse katvuse teostatavuse 65,6% enne heuristilist järeltöötlust. Juhendatud õppe piirangute lahendamiseks rakendati näitleja-kriitiku RL-mudelit, mis näitab paremat tühimike suhet 1,42. Kavandatud kaheastmeline hübriidne ML-RL arhitektuur, mis ühendab juhendatud servast veoautoni määramised RL-põhise marsruutimisega, saavutab parema töökoormuse tasakaalu variatsioonikordaja 0,843 võrreldes 0,903-ga juhendatud ja 0,945-ga RL-i puhul ning parema esialgse katvuse teostatavuse 70,3%, kuid veidi kõrgema tühimike suhte 1,53, mis toob esile kompromisse tegevuse efektiivsuse ja töökoormuse jaotuse vahel. Neid lähenemisviise valideeriti Tartu linnas, Eestis, reaalsetes teedevõrkudes läbi viidud katsetega. Tulemused näitavad, et puhas RL pakub kõige tõhusamaid lahendusi marsruudi pikkuse ja tühimike osas, samas kui hübriidne ML-RL mudel näitas tugevusi töökoormuse võrdsuse ja katvuse teostatavuse osas.

Võtmesõnad: Kaarmarsruutimise probleem, lumelükkamise marsruutimine, segatud täisarvuline lineaarprogrammeerimine, graafi närvivõrgud, tugevdusõpe, hübriidne ML-RL mudel

CERCS: P170 - Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine; P176 - Tehisintellekt

1. Acknowledgement

My sincere gratitude goes to my supervisors, Dr. Kallol Roy and Dr. Jaan Übi. Their keen insights, timely advice, and unwavering support guided this thesis from idea to completion. I am also grateful to the University of Tartu Institute of Computer Science staff and my fellow researchers for their practical help and stimulating discussions. Finally, heartfelt thanks to my family and friends for their unwavering support.

Contents

2. Introduction

2.1 Problem Definition and Motivation

The Arc Routing Problem (ARP) is a fundamental combinatorial optimization challenge in operations research, where vehicles are routed to cover a subset of roads or edges within a network, typically to minimize total travel cost, distance, or time while ensuring that each edge in the target subset is traversed exactly once [1]. Unlike the node-centric Vehicle Routing Problem (VRP), ARPs explicitly require servicing edges, making them particularly relevant in street-level operations scenarios. ARPs are critical for real-world applications like urban planning, logistics, and service maintenance. Some practical examples include garbage collection, where garbage trucks must traverse routes to efficiently service all streets in a district, reducing fuel costs and environmental impact [2]. Another practical application of ARP is in post-delivery and street sweeps, where optimal coverage is required to deliver mail and maintain balanced workloads among vehicles [3]. One more practical application of ARP is in snowplow routing, which will be the focus of this research. In snowplow routing, snow trucks that need to clear snow from roads must be routed efficiently and evenly to ensure road mobility and public safety during winter storms [4]. These applications underscore the need for precise routing solutions that minimize operational costs, reduce service times, and improve overall service quality. Figure 1 shows an example visualization of an Arc Routing Problem scenario.

Classical methods for solving ARPs commonly include exact approaches, such as Mixed Integer Linear Programming (MILP) [4], and heuristic algorithms such as Genetic Algorithms, Tabu Search, and Local Search techniques [5]. Although exact methods like MILP guarantee optimal solutions, they suffer from considerable computational overhead, making them impractical for large-scale, real-time routing decisions [6]. In contrast, heuristic methods provide computationally efficient solutions, but often fail to guarantee complete coverage of all required edges, leaving crucial tasks unaddressed and violating connectivity constraints [7]. Emerging trends in recent years have seen growing interest in Machine Learning (ML) and particularly Reinforcement Learning (RL) techniques for ARPs, promising scalability, adaptability, and rapid inference suitable for real-world deployments [8]. However, current ML-based approaches frequently encounter some critical issues. More often than not, the routes predicted by the ML models omit crucial service edges and lead to incomplete service coverage [8]. Furthermore, ML models can produce fragmented routes that violate connectivity and are infeasible in practice without additional manual adjustments. These limitations highlight the need to advance

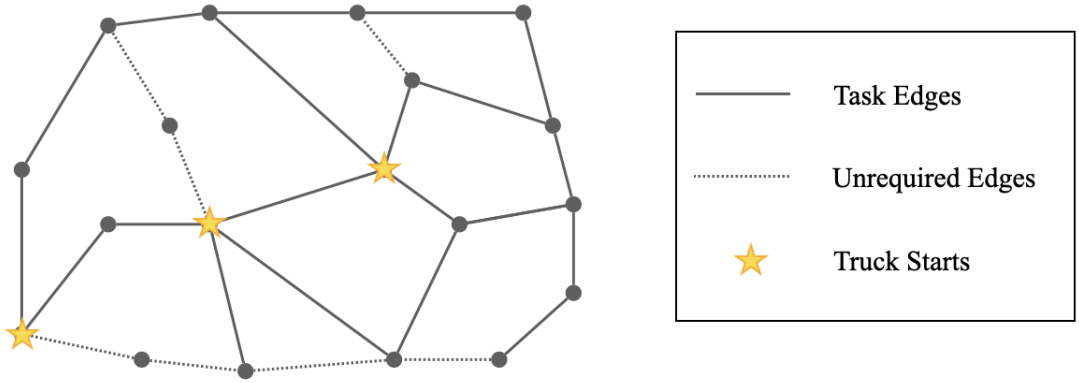

Figure 1. Example visualization of an Arc Routing Problem scenario illustrating required task edges, unrequired edges, and truck start points.

ML-driven ARP solutions, which inherently ensure complete edge coverage, connectivity, and feasibility, particularly in challenging scenarios such as snow removal in rural or urban environments.

In this paper, we address the following key questions:

1. How effectively can operations research methods, specifically Mixed Integer Linear Programming (MILP), solve the Snowplow Arc Routing Problem (SARP)?
2. Can a supervised learning approach reliably approximate MILP-based optimal solutions while guaranteeing complete edge coverage and route connectivity?
3. How can reinforcement learning (RL) techniques be effectively integrated into an ML framework to enhance routing performance and overcome connectivity and coverage issues inherent in current ML approaches?
4. What are the comparative strengths and weaknesses of the classical ML, RL, and hybrid ML-RL approaches compared to the classical MILP solutions in terms of scalability and solution quality?

With these questions in place, the main objective of this thesis is to develop, implement and evaluate various models and approaches for optimizing snowplow truck routes to build a robust and practically applicable framework that reliably solves the arc routing problem adapted to road networks in Estonia. An MILP model is formulated to precisely optimize snowplowing tasks to serve as ground truth and a benchmark. Next, a supervised ML model is developed for truck route planning leveraging graph neural networks (GNNs). An actor-critic reinforcement learning technique is also implemented. Finally, a hybrid model that integrates supervised

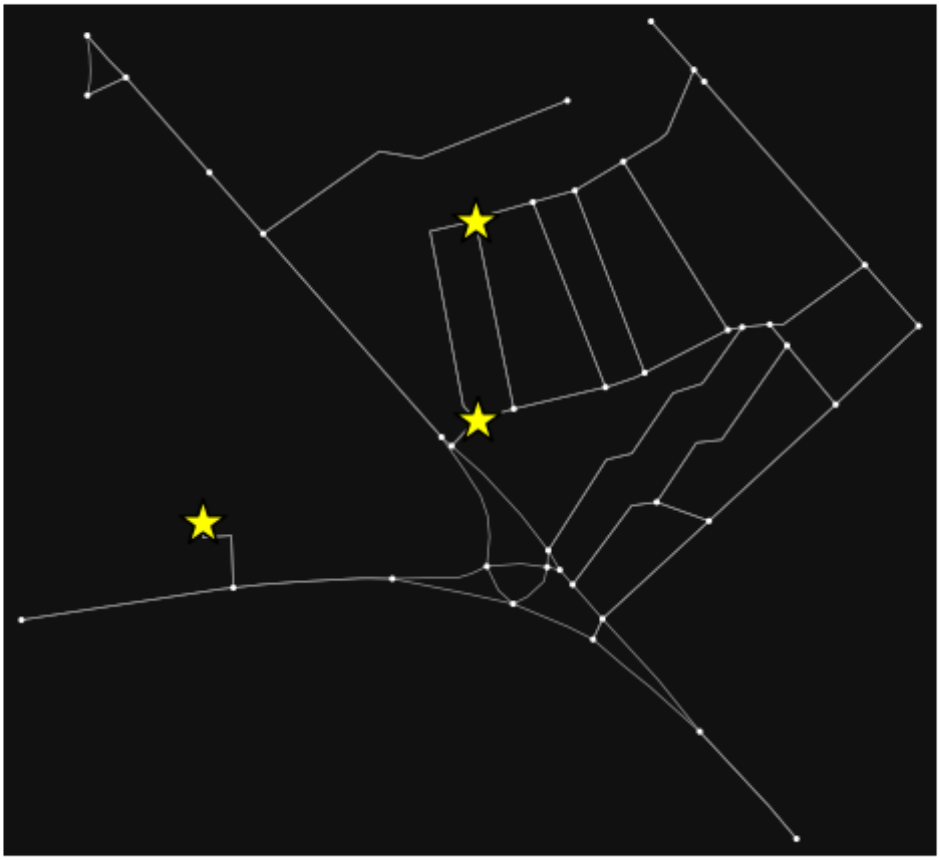

Figure 2. Tartu, Estonia road network example showing start positions (depots) of snowplow trucks.

ML with reinforcement learning (RL) is implemented to dynamically and effectively plan routes, minimizing total deadhead distances and balancing workload across multiple vehicles. All of these models' performances are validated against the exact MILP solution, comparing improvements, and particularly looking at edge coverage, connectivity constraints, and workload balance while evaluating the practical applicability and efficiency of the models through extensive computational experiments on real-world road datasets from Estonia, as shown in Figure 2.

2.2 Thesis Structure Overview

This chapter introduced the arc routing problem, the motivation of the research, the questions, the objectives, and the scope. The rest of the thesis document is structured as follows. Chapter 2 presents a background and review of the existing literature on ARP solutions, traditional optimization methods, and machine learning approaches, identifying the critical gaps that this research addresses. Chapter 3 describes the methodology used, highlighting the OpenStreetMap

dataset creation process, model formulation, and the three methodological approaches used in this research: Mixed Integer Linear Programming (MILP) Optimization, supervised learning, and reinforcement learning. Chapter 4 gives the experiment setup and implementation details and a comprehensive view of the data preparation, network encoding, model architectures, training procedures, and evaluation metrics. Chapter 5 presents comparative experimental results that demonstrate the effectiveness and efficiency of the developed models against benchmarks with detailed statistical and visual analyzes. Chapter 6 discusses and interprets the results, highlighting the contributions of the implemented methods, their practical relevance, and their limitations. Finally, the paper concludes with Chapter 7, summarizing significant findings, contributions, and potential directions for future research.

3. Background and Literature Review

This chapter comprehensively reviews the existing literature on Arc Routing Problems (ARPs), highlighting traditional optimization approaches, heuristic techniques, and modern machine learning methods with particular attention to snowplow routing scenarios. The chapter concludes with a discussion of reinforcement learning applications in arc routing.

3.1 Arc Routing Problems (ARPs)

Arc Routing Problems (ARPs) involve determining optimal routes that traverse certain specified arcs or edges within a graph exactly once, with the objective of minimizing total travel distance or cost [1]. Two primary variants of ARPs frequently discussed in the literature are the Chinese Postman Problem (CPP), where all edges of a network must be covered at minimal cost, and the Rural Postman Problem (RPP), which requires servicing only a subset of edges [9]. In practical urban scenarios, ARPs frequently arise in applications such as trash sweeping and garbage collection [2]. ARPs are inherently NP-hard, making exact solutions computationally challenging for large-scale instances [3]. Some key challenges identified in the literature for ARPs include:

- **Complete Coverage**: This involves ensuring that all designated edges are covered exactly once, without omissions [1].
- **Deadheading**: This involves minimizing cases where serviced edges are unnecessarily traversed without performing required tasks. Deadheads significantly impact operational costs [4].
- **Workload Balancing**: This involves distributing tasks evenly among vehicles to avoid excessive workload disparities [5].
- **Connectivity Constraints**: This involves preventing disconnected routes to ensure feasible and viable routes [10].

3.2 Snowplow Routing

Snowplow Routing presents unique challenges for ARP, due to unpredictable weather conditions and resource constraints [7]. Snow plowing routing involves determining efficient paths for plowing and salting roads, requiring routes to be optimized for distance and balance in workload among available trucks [4]. Hajibabai et al. [4] highlight complexities such as road plowing priorities based on road importance, resource replenishment such as salt supply,

and managing real-time disruptions like traffic or road closures. Perrier et al. [7] provide a comprehensive survey of existing methodologies that specifically target winter maintenance, noting the importance of systematic modeling to handle task prioritization, replenishment, and operational constraints. Common practical issues in snowplow routing include prioritization of roads where roads with critical infrastructure or high traffic volumes must be serviced first, dynamic operational conditions due to sudden changes in weather variability or equipment failures, and limited U-turn locations where heavy vehicle maneuverability is restricted [4]. These challenges complicate snow truck route planning and emphasize the need for dynamic and robust solutions capable of quick real-time adjustments.

3.3 Optimization Techniques for Routing Problems

Classical approaches to ARP fall mainly into two categories: exact methods and heuristic/metaheuristic solutions [11]. Mixed Integer Linear Programming (MILP) provides optimal solutions by formulating routing problems through mathematical optimization frameworks [4]. MILP models guarantee global optimality, which is crucial for validating heuristic and ML-based approaches. However, these solutions often become computationally infeasible with increasing network size or complexity due to exponential scaling [5]. Given these limitations, researchers have frequently resorted to heuristic and metaheuristic algorithms like Greedy Heuristics, which provide quick, intuitive solutions without guaranteeing optimality [12], Genetic Algorithms (GA) that use evolution-inspired approaches exploring large search spaces through iterative selection and mutation, producing near-optimal results [10], and Tabu Search and Simulated Annealing methods that involve iterative improvement with mechanisms for escaping local optima, shown effective in numerous ARP scenarios [13]. Although these heuristic solutions are computationally efficient, they often sacrifice optimality, not consistently guaranteeing coverage completeness or optimal connectivity [13].

3.4 Machine Learning Approaches for Routing Problems

Recent advances in machine learning (ML), particularly Graph Neural Networks (GNNs), have significantly impacted combinatorial optimization and vehicle routing domains [14]. ML models have shown promise in scalability and adaptability, rapidly inferring high-quality solutions without the exhaustive search inherent to classical methods. Nazari et al. [8] introduced an end-to-end ML approach for the Vehicle Routing Problem using a sequence-to-sequence model based on attention mechanisms that demonstrated considerable efficiency gains compared to classical methods. Similarly, Kool et al. [15] proposed attention-based Transformer architectures,

highlighting the potential of ML to solve combinatorial problems effectively. Despite these advances, ML-based approaches commonly face specific challenges. One major challenge is that ML models often provide incomplete edge coverage, resulting in routes missing certain service edges due to implicit training biases or model limitations [16]. Another challenge of ML route predictions is the connectivity of the routes produced. ML solutions often generate disjointed routes or subtours, lacking explicit constraints inherently enforced by classical optimization methods [17]. Another challenge of ML solutions is the limited generalization of the solution. ML models trained on specific datasets might underperform when faced with significantly different network topologies or varying operational conditions. These issues illustrate the need to integrate traditional optimization constraints and domain knowledge into ML-driven frameworks to guide the ML approaches towards more robust solutions.

3.5 Reinforcement Learning in Vehicle Routing

Reinforcement Learning (RL) is a subfield of ML that has recently emerged as a promising alternative to routing optimization. RL models learn routing strategies through interactions with simulated or real-world environments, dynamically adjusting routes based on immediate feedback rewards [18]. RL's usefulness stems from its capacity to adapt to dynamic scenarios, making it highly appealing for real-time vehicle routing problems like snowplow routing. Kool et al. [15] and Nazari et al. [8] demonstrate the capability of RL to manage large-scale routing problems with real-time decision-making requirements. However, RL approaches also have some notable challenges. For example, ensuring task coverage and maintaining route connectivity without explicit constraints are common issues of RL. Another common issue associated with RL methods is with sparse rewards. Efficiently structuring rewards to ensure complete coverage of tasks and minimize deadheading is complex [18]. Furthermore, RL agents can create disconnected routes if constraints are not explicitly modeled in the environment or the reward system; thus, maintaining connectivity is hard. Furthermore, RL models typically require extensive training episodes, posing computational resource challenges for real-world applications [19]. Addressing these challenges requires carefully designed RL frameworks that explicitly incorporate domain-specific constraints, such as complete edge coverage, connectivity, and workload balance, directly into their reward structures and environment interactions. Table 1 presents a comparative summary of the methods discussed, showing their strengths and limitations.

Table 1. Comparative summary of ARP methodologies | exact (MILP), heuristic/metaheuristic, ML, and RL.

Technique	Description	Strengths	Limitations
Exact (MILP)	Mathematical optimization using integer programming formulations to find optimal routes	• Guarantees globally optimal solutions. • Clearly defined constraints and route feasibility.	• Computationally intensive; not scalable. • Infeasible for large or real time routing problems
Heuristic/ Metaheuristic	Approximation methods such as Genetic Algorithms, Tabu Search, and Simulated Annealing	• Computationally efficient; scalable for large networks. • Often produces near-optimal solutions quickly.	• No guarantee of optimality. • Risk of incomplete, disconnected solutions.
Machine Learning (Supervised)	Data-driven predictive methods (e.g., Graph Neural Networks) trained to approximate optimal routes	• Fast inference; suitable for real time scenarios. • Learns complex patterns directly from data. • Good scalability	• Requires extensive labelled datasets for training. • Potential issues with incomplete task coverage and connectivity constraints.
Reinforcement Learning (RL)	Methods that learn routing policies through environment interaction, and optimizing via rewards	• Adaptable to dynamic, real time environments. • Learns complex, non linear policies directly from interactions. • Good scalability and adaptability.	• Requires careful reward shaping to avoid sparse reward issues. • Can lead to disconnected routes or incomplete coverage • High computational training cost.

In general, the reviewed literature shows that there is limited research on integrating traditional optimization constraints into ML and RL frameworks to guarantee complete service coverage and connectivity. There are also insufficient hybrid methodologies that combine supervised learning for initial assignments and reinforcement learning for dynamic routing optimization. This thesis aims to bridge these gaps by developing a robust hybrid ML-RL framework that ensures comprehensive edge coverage, connectivity, and efficiency, which is systematically validated against optimal MILP benchmarks.

4. Methodology

This chapter outlines the methodological framework and approaches used to solve the Snowplow Arc Routing Problem (SARP)on road networks. The chapter describes data collected from OpenStreetMap using OSMnx, followed by a detailed mathematical formulation of the routing problem. It then elaborates on three primary approaches, Mixed Integer Linear Programming (MILP), supervised learning, and reinforcement learning, and introduces the hybrid two-stage ML-RL model design and how the limitations identified in previous chapters were addressed.

4.1 Data Collection

Data for this research are obtained from OpenStreetMap (OSM), an open source community-maintained global mapping project that provides rich and comprehensive geographic data suitable for network-based routing problems [20]. The OSMnx Python library is developed by Boeing [21] extracts, processes, and visualizes street networks. As the problem focuses on snow plowing in Estonia, the Tartu network was specifically chosen. This network has a good combination of urban road structures and therefore provides a realistic representation of the challenges faced by snow removal services. Using OSMnx, the road networks were downloaded based on geographic queries and the extracted data included essential attributes such as the geographic coordinates (latitude, longitude) of the nodes, the geometry of the edges, and the lengths of each route. The networks are pre-processed to remove isolated nodes and small disconnected subgraphs, ensuring a coherent and feasible network for routing tasks.

4.2 Mixed Integer Linear Programming (MILP) Formulation

The Snowplow Arc Routing Problem (SARP) is formally defined as a directed arc routing problem, represented as a directed graph $G = (V, E)$ where:

- V is the set of nodes that represent intersections on the road.
- E is the set of edges (u, v) that represent road segments that require snow removal, also known as the task edges, and u and $v \in V$.

The purpose of the SARP is to determine a set of routes for a fleet of snowplow trucks, each starting and ending at its designated depot, so that each road segment in a specified subgraph is plowed exactly once. Figure 2 shows an example of the SARP on a section of the Tartu Road Network. With trucks potentially starting at multiple depots, the model defines the variables below.

- $K = \{1, \ldots, m\}$ is the set of snow plow trucks whose routes need to be planned.
- $T = \{1, \ldots, |T|\}$ is the set of task edges that must be serviced. Each element in T is a directed road segment (u, v) in the Graph G. Since in this formulation, all edges need to be plowed, $T = E$.
- $D = \{d_k | k \in K\} \in V$ is the set of individual depot nodes with each element in D representing the depot/start node of the truck.

The formulation also defines the following parameters.

- p_{ij} is the cost associated with the edge (i, j). This is length of the edge $(i, j) \in E$
- d_{ij} is deadhead cost for when a truck k directly moves from node i to node j. This cumulative deadhead distance will give the distance traveled by a truck without performing a plowing task.
- M is a large constant that is intentionally set to be larger than the longest possible route to allow for subtour elimination constraints and depot-to-first-task initialization constraints to turn off when their associated binary variable is 0, as seen in constraints 6 and 7 below.
- $c_{\text{LOS}} \geq 0$ is the weight on the longest route term in the objective function to control workload balance of the trucks.

The distances are precomputed with Dijkstra on the graph that the task edges are extracted from OpenStreetMap

4.2.1 Decision Variables

The model uses three decision variables. It uses binary variables x_{ijk} to determine both the assignment of task edges to trucks and the precise sequencing of travel between nodes. x_{ijk} is assigned 1 if the truck k is scheduled to traverse the edge (i, j) and 0 if otherwise. The second variable u_{vk} is used to record the cumulative distance accumulated by the truck k the moment it first reaches the node v. The third variable z is a continuous one that captures the upper bound on the total length of travel of each truck.

Table 2 summarizes the decision variables for the MILP model.

Table 2. Decision variables of the MILP model.

Variable	Domain	Meaning
x_{ijk}	$\{0, 1\}$	Equals 1 if and only if truck k traverses edge (i, j)
u_{vk}	$R_{\geq 0}$	Cumulative distance when truck k first arrives at node v.
z	$R_{\geq 0}$	Upper bound on the longest individual deadhead distance.

4.2.2 Objective Function

With these variables and parameters in place, the optimization objective is to minimize the sum of all deadhead distances and the longest individual route distance to ensure the balance of vehicle workload.

Formally, the objective function is given below.

$$\min \sum_{k \in K} \sum_{(i,j) \in E} d_{ij}\, x_{ijk} \;+\; c_{\text{LOS}}\, z \tag{1}$$

4.2.3 Model Constraints

The MILP formulation is constrained as follows. For every truck, the sum of the edges that leave its depot must be 1 as seen in constraint (2). This constraint forces each vehicle to start a route instead of remaining idle. Similarly, at least one edge must enter the depot, also forcing every truck to come back at the end of its tour. Trucks are allowed to make multiple visits back to the depot because the model assumes that the truck may need to use the depot to visit other potential edges.

$$\sum_{(d_k,j) \in \delta^+(d_k)} x_{d_k jk} \;\geq 1, \qquad \forall k \in K \tag{2}$$

$$\sum_{(i,d_k) \in \delta^-(d_k)} x_{i d_k k} \;\geq 1, \qquad \forall k \in K \tag{3}$$

As seen in constraint (4), at every non-depot node, the incoming and outgoing edge counts must balance for each truck, ensuring continuous, connected paths rather than disconnected fragments. This also ensures that a truck does not end its route at a node that is not a depot.

$$\sum_{(v,j) \in \delta^+(v)} x_{vjk} - \sum_{(j,v) \in \delta^-(v)} x_{jvk} = 0, \quad \forall v \in V \setminus \{d_k\},\ \forall k \in K \tag{4}$$

Constraint (5) ensures that each task edge is plowed once and only once since the binary variables of all trucks sum to exactly 1 on every required edge segment.

$$\sum_{k \in K} x_{uvk} \;=\; 1, \quad \forall (u,v) \in T \tag{5}$$

Constraint (6) creates a relation between the arrival time variables u_{ik} and u_{jk} when an edge is selected. With this inequality, a cumulative distance is propagated so that subtours that do not include the depot are avoided.

$$u_{ik} + p_{ij} + d_{ij} \;\leq\; u_{jk} + M\,(1 - x_{ijk}), \quad \forall k \in K,\ \forall (i,j) \in E,\ i \neq j \tag{6}$$

Constraint (7) initializes the u variable for when trucks first leave their depot, thus keeping the arrival time to nodes chronologically consistent from the beginning.

$$u_{jk} \;\geq\; d_{d_k j} - M\,(1 - x_{d_k jk}), \quad \forall k \in K,\ \forall (d_k, j) \in \delta^{+}(d_k) \tag{7}$$

The total distance traveled by each truck is capped by the continuous variable z as seen in constraint (8) for the objective function to minimize and balance workload across the fleet.

$$\sum_{(i,j) \in E} d_{ij}\, x_{ijk} \;\leq\; z, \quad \forall k \in K \tag{8}$$

Since u_{vk} and z represent distances traveled, they are restricted to be non-negative. The binary variable x_{ijk} is then restricted to values 0 or 1 to indicate edge usage.

$$x_{ijk} \in \{0,1\}, \qquad u_{vk} \geq 0, \qquad z \geq 0 \tag{9}$$

Using the Mixed Integer Linear Programming (MILP) formulation above, the model was implemented using Gurobi, a high-performance optimization solver that guarantees global optimality [22]. The MILP formulation closely follows the mathematical formulation defined above. The primary reason for adopting the MILP approach is its ability to provide exact optimal solutions [23]. Given the computational intensity, the MILP approach serves primarily two purposes. First, this generates ground-truth route solutions that are used for supervised learning training datasets. Second, it provides benchmarks for the performance evaluation and validation of the ML and RL methods. As mentioned earlier, the Gurobi solver was configured to handle

subtour elimination, task coverage, and connectivity and provided precise and reliable results. However, this came at the cost of computational scalability limitations for larger networks. As such, the MILP solution was limited only to two and three trucks, as the more trucks you have, the more constraints the model has to work with, thus increasing the needed computational power.

4.3 Supervised Learning Approach

Using the data obtained from the MILP implementation, a supervised learning model is set up to predict the truck routes for each given graph G and the required task edges. The model first performs an assignment of every required task edge to one of the trucks, and for each truck, an ordered sequence of vertices that begins and ends at the truck's depot is predicted to traverse all of its assigned task edges at least once.

4.3.1 High-Level Model Overview

At the core of the supervised model is the representation of the task-edge subgraph as a line graph $L(G_{\text{task}})$, so that each task-edge becomes a node. A Graph Convolutional Network [24] is then used to predict a categorical distribution over the trucks for every such node, thereby learning how to partition the service workload. From this initial assignment, a transformer-based pointer network greedily decodes a tour for each truck. The decoding is interleaved across trucks and is governed by hard connectivity and coverage masks, ensuring that only topologically feasible next moves are considered. Each stage uses a graph encoder that provides task-specific node embeddings and a graph-level context vector. The parameters are then trained end-to-end under supervision from ground truth samples obtained from the MILP formulation.

4.3.2 Representation of the Task Graphs

The supervised ML pipeline uses a two-layer edge-conditioned GNN that operates on the task-edge subgraph $G_{\text{task}} \subseteq G$. Each vertex of G_{task} is annotated with a five-dimensional feature vector designed to capture the accessibility of the depot and the structure of the graph. The features include:

- **Normalized distance to nearest depot**: This feature encourages early attention to accessible streets.
- **Normalized degree**: This feature is used to distinguish intersections from dead ends.

- **Normalised incident-edge-length sum**: This feature presents insights into the cost associations of road intersections.
- **Clustering coefficient** [25]: This feature measures the availability of short cycles in the network
- **Normalised closeness centrality** [26]: This feature encourages early attention to reachable streets.

The normalized physical lengths $\ell(u, v)/\max_{e \in E_{\text{task}}} \ell(e)$ of edges are also fed to the model as attributes. Using NN-Conv [27], the two Edge-Conditioned Convolution layers learn message weights as a function of these attributes and produce node embeddings $\mathbf{h} \in R^{N \times d}$ and a graph embedding $\mathbf{g} \in R^d$ via global mean pooling with d as the final embedding dimensionality. This shared representation of the task subgraph serves as input features to the edge-assignment GNN and as memory keys for the pointer decoder.

Given how the line graph, $L(G_{\text{task}})$, is structured, linking pairs of task edges that share an endpoint allows the network to capture if two edges incident to the same vertex are allocated to different trucks. This helps the models learn which tours to cross. A stack of GraphConv layers with ReLU activations and dropout propagates interaction data, and a final linear classifier outputs logits over the number of truck classes for each edge.

4.3.3 Interleaved Transformer Pointer Decoder

During the decode phase, for each truck k, the decoder must predict an index sequence in which every consecutive node pair (i_{t-1}, i_t) of the sequence forms a task edge previously assigned to k with a special EOS symbol that signifies the return to the depot. The candidate keys for prediction are the concatenation of node embeddings and a learned EOS vector. A shared start token, representing the node the truck starts on, indicates the beginning of each sequence. The model uses a multilayer Transformer decoder with sinusoidal positional encoding and attention to the fixed memory of candidate embeddings. Self-attention is used to capture longer-range ordering constraints, such as avoiding early depot returns. At the same time, cross-attention queries the candidate set to emit a pointer distribution at every step. The model is forced to use greedy inference to respect two constraints: connectivity and coverage mask. Only neighboring vertices that are incident to an uncovered task edge from the current vertex are eligible for connectivity selection. For coverage and to ensure a complete return to the depot, once all assigned edges have been serviced, all trucks are brought back to the depot using the shortest

route. Decoding is interleaved across trucks, where truck 1 makes the first step, then truck 2, and so on, cycling until all trucks have terminated to encourage implicit coordination and allow one truck's early decisions to influence masking for others.

The model computes a composite loss to train. Since historical optimal solutions specifying ground-truth assignments and tours are available, the edge-assignment loss and the pointer sequence loss are combined. The edge assignment loss is given as the cross-entropy between predicted logits and one-hot truck labels, and the pointer sequence loss is given as the token-level cross-entropy plus a coverage penalty that adds a constant cost for every task edge left unvisited in the predicted sequences. This loss guides learning toward both token accuracy and completeness of coverage. The composite loss can be expressed as

$$L = L_{\text{assign}} + (L_{\text{seq}} + {}_{\text{cov}} L_{\text{cov}}), where \alpha and \lambda_{\text{cov}} \tag{10}$$

balances the relative influence of assignment and routing quality. Gradients flow through all stages, allowing the encoder to learn representations that are helpful simultaneously for partitioning and tour construction. Figure 5 presents a level outlook on the supervised learning architecture used to predict to solve the snowplow optimization problem.

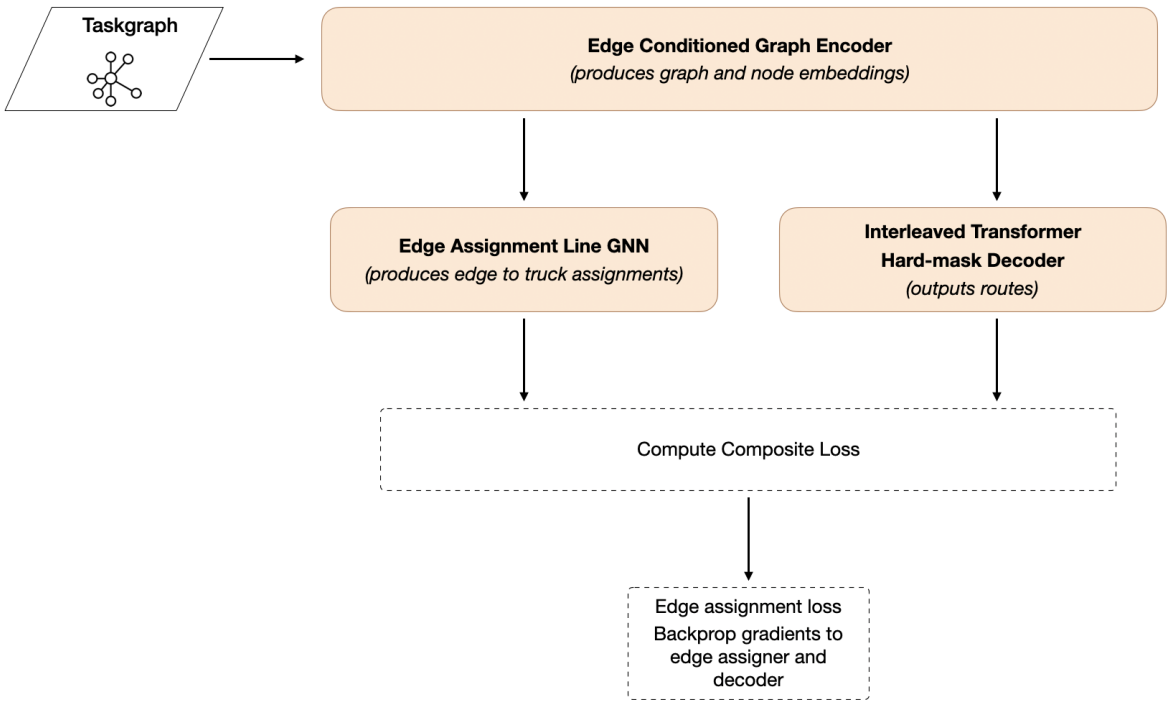

Figure 3. A High-level Summary of the Supervised ML Architecture used for Snow Route Optimization

4.4 Reinforcement Learning Approach

4.4.1 High Level Architecture

The RL model uses the same graph representation and encoder as the supervised learning model. For its decoder, it uses a Transformer-based Pointer Network too. This was adopted due to its proven effectiveness in sequence-to-sequence optimization problems [16]. Since no prior edge assignments are done here, in this single-step graph-to-route approach, the entire set of road segment tasks is embedded once and decoded into up to M truck tours by a learned policy. The architecture, illustrated conceptually in **??**, combines an edge-conditioned graph neural encoder that transforms the task subgraph into dense latent representations, a transformer-based pointer decoder that generates the route of each truck under hard combinatorial constraints, and a state value critic that supports a reduction in actor rating variance.

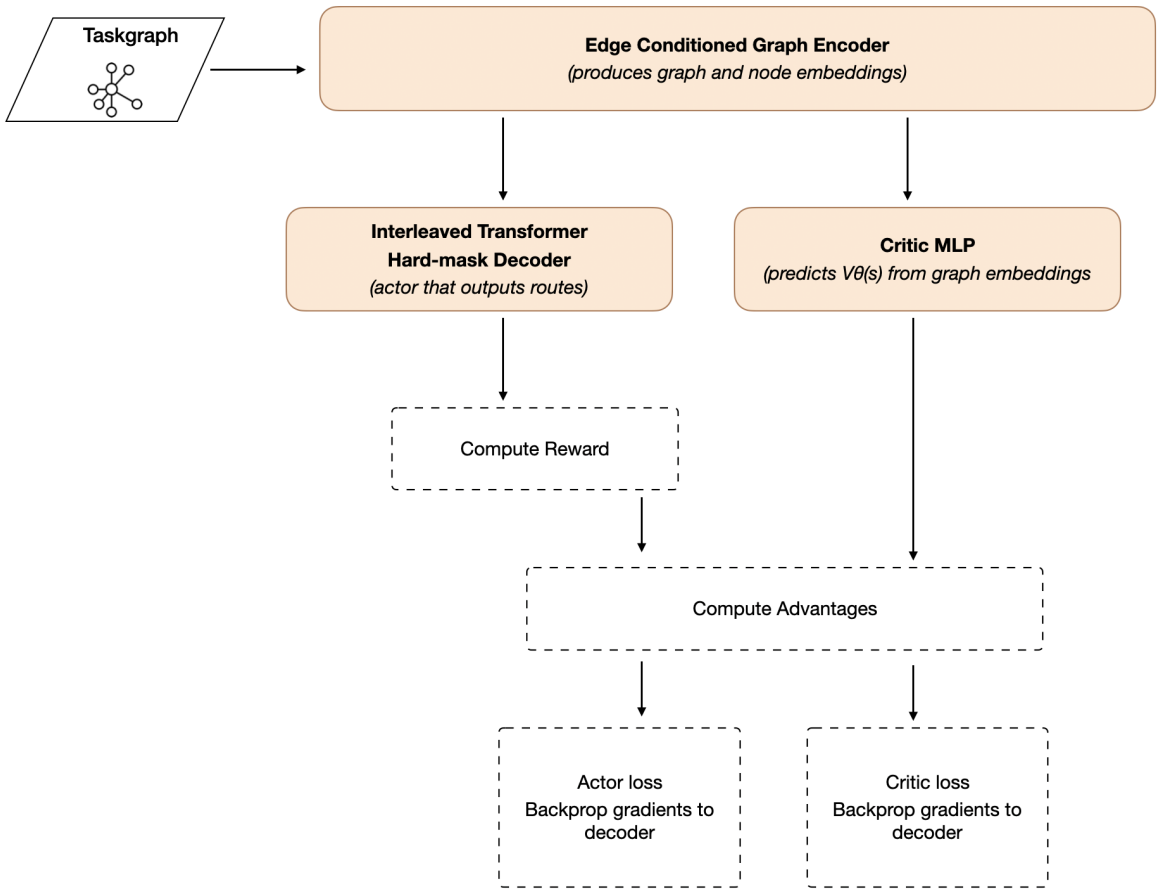

Figure 4. A High-level Summary of the Reinforcement Learning Architecture used for Snow Route Optimization

4.4.2 Actor-Critic Architecture for RL

For faster convergence, a critic network approximates the state value $V_\theta(s)$.

$$V_\theta(s) = \text{MLP}(g), \qquad \text{MLP} : R^d \to R \tag{11}$$

where s is the state presented to the agent. In this problem, a state s is the entire task-edge sub-instance currently being routed. The variable g is the global embedding of the encoder. It is produced by the edge-conditioned encoder via global mean pooling over node embeddings. It is a fixed-length permutation invariant vector summarizing s. A single hidden layer then linearly projects the graph embedding with non-linearity injected into the embeddings via ReLU. MLP(θ) is a two-layer fully connected network with ReLU nonlinearity and parameters that maps the graph embedding to a scalar. The state-value estimate $V_\theta(s)$ is the critic's prediction of the expected cumulative reward that the current policy will obtain from state s onward. It serves as a baseline to reduce variance in the policy gradient update. θ are the trainable weights of the critic's MLP. They are not shared with the actor decoder. The policy updates then follow the advantage-weighted objective

$$\nabla_\phi J(\phi) = E\big[(R - V_\theta)\nabla_\phi \log \pi_\phi(a|s)\big] \tag{12}$$

which exhibits lower variance than REINFORCE [28] with a running scalar baseline.

$J(\phi)$ is the expected return (objective) under the current stochastic policy π_ϕ. The variable ϕ represents the trainable weights of the actor which are all parameters of the pointer decoder that produce action probabilities. $E\big[.\big]$ is the expectation over the trajectory distribution induced by π_ϕ in the environment. R is the episode reward obtained after executing the complete set of truck routes decoded in one forward pass. More details of the reward, R can be seen further below. $V_\theta(s)$ is the baseline prediction from the critic seen above. Subtracting it from the reward R forms the advantage $A = R - V_\theta(s)$. When $V_\theta(s)$ is accurate, A has zero mean, which reduces the gradient variance without changing the optimum. The variable a is one concrete action sampled during training. For the pointer decoder, an action is the next node index chosen at a particular decoding step. In the end, one full episode comprises many such actions. $\pi_\phi(a|s)$ is the probability that the actor selects the action a in state s. Finally, $\nabla_\phi \log \pi_\phi(a|s)$ is the score function of the selected action. This function scales the gradient update in proportion to how

surprising the action was under current policy. With this, the update reinforces log-probabilities of actions that led to higher-than-expected returns and suppresses those that underperformed.

4.4.3 Reward Formulation

The scalar reward signal used in the model is as follows:

$$R = w_{\text{cov}} |\text{Covered}| - w_{\text{miss}} |\text{Missed}| - w_{\text{dh}} \text{Deadhead} - w_{\text{long}} \max \text{Length} - w_{\text{ret}} \text{ReturnPenalty} \tag{13}$$

|Covered is the number of required task-edges plowed by any truck. |Missed is the task edges that remain unserved at the end of the episode. The term is subtracted to impose a strong penalty on infeasible solutions. Deadhead is the aggregate distance traveled on non-task road segments while moving between required edges. Minimizing deadhead improves service time. max Length is the length of the longest individual truck route. By penalizing this value, the model encourages a balanced workload among the trucks. ReturnPenalty is the distance a truck drives after its final serviced edge to return to its depot. This term keeps routes compact and encourages the model to finish closer to the starting points of the trucks. The variable w_* is the respective scalar weights that trade-off the five objectives. In general, a larger reward value is desirable, so its coefficient is positive. Coverage and missed-task terms strongly dominate, steering learning toward feasible, complete solutions. Penalties are also added to discourage excessive deadheading, unbalanced workload, and late depot returns.

By coupling a differentiable encoder with an autoregressive pointer policy, gradients propagate from route-level rewards back to the graph features without intermediate heuristic stages. Similarly to the Supervised ML architecture, hard masks encourage feasibility in every decoding step to minimize the need for extensive, expensive repair heuristics and ensure that the critic learns policy value estimates for realistic connected routes.

4.5 Two-Stage Hybrid ML-RL Design

Combining the strengths and limitations of the individual methodologies described above, this research explores a two-stage hybrid ML-RL model design, which effectively integrates supervised learning with reinforcement learning. The two-stage hybrid learning architecture couples supervised graph representation learning with an actor-critic reinforcement learning (RL) procedure. The main objective of this design is to take advantage of the benefits presented by the supervised ML model and incorporate those into the designed RL technique. This design helps to decompose the arc routing problem into two tractable sub-decisions: edge-to-truck

assignment and route construction, and allows learning components to specialize in a narrower prediction task while still being trained end-to-end within a single pipeline.

The edge-to-truck assignment on the line graph uses the same model architecture described for the supervised learning model, with similar interleaved Pointer-Transformer route construction as both the supervised ML and the prior RL models.

For this hybrid learning strategy, the GNN assignment is first optimized with cross-entropy, providing a strong initial partition that accelerates downstream RL by narrowing the search space. The assigner is frozen, and the pointer transformer becomes the actor in an advantage-based RL scheme, similar to the one previously described in the RL-only model. An MLP critic maps the global graph embedding to a scalar state value estimate, stabilizing policy updates, and eliminating the need for a manually tuned running average baseline. Reward formulation, advantage estimation, and optimization follow the same design as described above for the RL.

Decoupling assignment and routing to specific, specialized models allows for targeted learning signals. The architecture allows the discrete output of the assigner to condition the masking logic of the decoder, but does not backpropagate the gradients to the assigner. The vector of edge-to-truck assignments produced in the first stage is transformed into per-truck service sets that are used directly to check for early-termination check of decoding and masking actions. Additionally, any edge left untraversed by its designated truck reduces the coverage term and lowers the reward. In contrast, shared graph embeddings ensure that both stages take advantage of a common learned representation of the input subgraph, fostering global coherence between decisions. The hybrid ML-RL architecture is summarized in Figure 5.

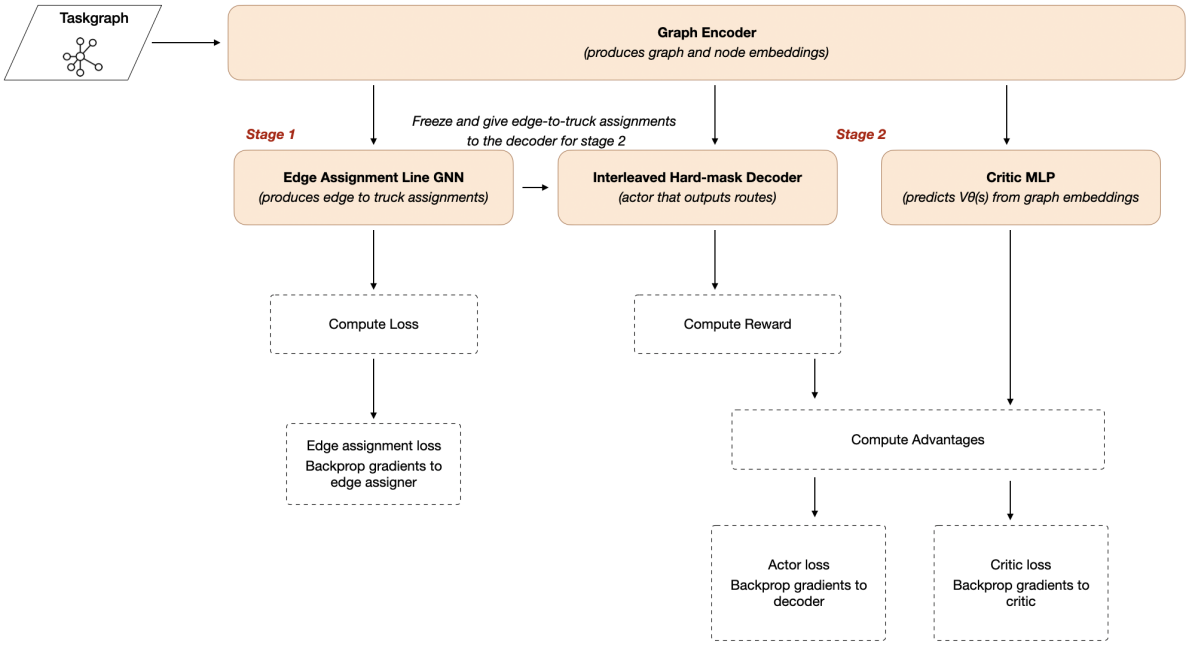

Figure 5. A High-level Summary of the Two-Stage Hybrid ML-RL Architecture used for Snow Route Optimization

5. Experimental Setup and Details

This chapter provides comprehensive details on the implementation of the methodologies introduced in Chapter 3 and the experimental setup. It begins with a description of the dataset generation process, including the data preprocessing and feature engineering steps. Then it describes the specific architectures used for Graph Neural Networks (GNNs), Transformer-based pointer networks, and reinforcement learning (RL) policy design. Subsequently, the chapter details the training procedures for the supervised learning and RL phases, concluding with a clear overview of the evaluation metrics used to validate model performance.

5.1 Ground Truth Dataset Generation via MILP

Following the MILP formulation described in Chapter 3, the MILP model was implemented in Gurobi **??**, and ground truth data sets were generated for optimal edge-to-truck assignments to provide exact sequences of the edges traversed by each truck. These not only served as benchmarks for evaluating the supervised and reinforcement learning methods implemented in this research, but they also served as comprehensive labels for training the supervised learning of edge assignments and optimal routing sequences.

5.1.1 Base Road Network Extraction and Subgraph Sampling

The base road network of Tartu, Estonia, was extracted using OpenStreetMap (OSM) using OSMnx, a Python package explicitly designed for street network analysis **??**. To diversify the topology of the generated data sets, a helper routine is used to create random subgraphs from the major Tartu road network to create a variety of subnets to serve as task graphs. The routine randomly draws a central node v^* in V and extracts all roads with a radius from the central node. The procedure is repeated to generate 1000 samples of unique subgraphs. These subgraphs were cleaned to remove isolated nodes, ensuring route feasibility and operational realism. For every subgraph, a further loop creates three variations of the sample to create distinct MILP instances, thereby exploring different vehicle fleet sizes and depot placements. Due to the limited computational complexities of the free version of the Gurobi library, the fleet size was limited to a maximum of 3 trucks with a minimum of 2 trucks. For each sample, K distinct depot nodes are selected that satisfy the $D = \{d_k | k \in K\} \in V$ variables from the MILP formulation in Chapter 3. All edges of the generated subgraphs are labeled as required task edges, so at the end $T = \{(u, v) | (u, v) \in E\}$. Their plowing cost, p_{ij}, is set to the geometric length of the edge. An auxiliary distance matrix for depot to task, and task to depot, and task to task deadhead distances are computed using Dijkstra on G to cache and use for further look-up

later. In the end, the look-up matrix gives all distances the model needs.

$$d_{i,j,k} = \begin{cases} d^{\text{dep}}_{k,j}, & \text{if } i = d_k, \\ d^{\text{task}}_{i,j}, & \text{if } i, j \in T, \\ d^{\text{dep}}_{k,i}, & \text{if } j = d_k. \end{cases}$$

$d^{\text{dep}}_{k,j}$ is the distance from depot k to node j $d^{\text{task}}_{i,j}$ is the deadhead distance between nodes i and j $d^{\text{dep}}_{k,i}$ is the distance from the node i back to the depot k.

5.1.2 MILP Construction

To solve the MILP formulation described in chapter 3, c_{LOS} in the objective function described in equation (1) is set to 1, and a described M value of 10^6 is set. With all these parameters and variables set, the model is allowed to optimize weeding out any subgraph variation that is unsolvable due to the size limit on the free-license version of the optimizer. Active edges for all sample instances that were optimally solved (indicated by a Gurobi optimizer status of $[(\text{status} = \texttt{GRB.OPTIMAL})]$) are extracted, and each truck route $[\big(d_k,\ t_{k,1},\ \ldots,\ t_{k,m_k},\ d_k\big)]$ is reconstructed into a linear sequence. The deadhead distance, plowing distance, and total distance traveled per truck are also computed and stored. A single seed also initializes both Python's random RNG and NumPy's RNG, ensuring determinism between runs and platforms. In the end, a CSV file consolidates all successfully solved variations with one record per variation, to serve as ground-truth data points and basis of evaluation for the learning models implemented in the research. 570 successfully solved samples were obtained. The schema of the CSV dataset file can be seen in Table 3.

Figures 6 and 8 show a visualization sample task graphs with 2 and trucks, respectively. Figures 7 and 9 show the optimal routing solutions obtained from the MILP model.

5.2 Supervised ML Model Implementation

This section specifies the concrete implementation decisions used in the study to implement the supervised ML model introduced in Chapter 3. All experiments were carried out on the same computational environment to ensure reproducibility. The supervised model takes into account the task edge subgraph, the truck start, the truck paths, the total distances of the truck, and the optimal objective value from the raw ground truth data given in CSV form. A task subgraph and a line graph representing this task subgraph are built as explained in Chapter 3.

Table 3. Schema of the ground truth dataset file

Column name	**Description of contents**
`Sample ID`	Unique label for the sub-graph instance (e.g., `subgraph_0`).
`num_nodes`	Cardinality $\lvert V \rvert$ of the sampled sub-graph.
`num_edges`	Cardinality $\lvert E \rvert$ (directed task edges) of the sub-graph.
`list_nodes`	Python-literal list of all node IDs in the sub-graph (unordered).
`list_edges`	List of ordered tuples (u, v) enumerating every task edge; ordering matches `edge_lengths`.
`edge_lengths`	Floating-point edge lengths (metres) one-to-one with `list_edges`.
`num_trucks`	Fleet size K for this variation.
`truck_starts`	List of depot node IDs $\{d_0, \ldots, d_{K-1}\}$.
`task_list`	Duplicate of `list_edges` stored for convenience.
`task_nodes`	Distinct node IDs that appear as end-points of any task edge.
`truck_paths`	Dictionary $\{k \mapsto \text{route}_k\}$ where each route is $\big[d_k, (e_1), (e_2), \ldots, d_k\big]$; depots are node IDs, serviced arcs are tuples.
`truck_total_distances`	Dictionary mapping each truck to its total distance (plowing + dead-head, meters).
`longest_total_distance`	Maximum value in `truck_total_distances`.
`longest_deadhead`	MILP variable z: largest dead-head distance of any truck (meters).
`truck_deadhead_distances`	Dictionary mapping each truck to its aggregate dead-head distance (meters).
`truck_deadhead_edges`	Dictionary mapping each truck to the sequence of edge pairs that constitute its deadhead segments.
`optimal_objective_value`	Optimal MILP deadhead value returned by Gurobi.

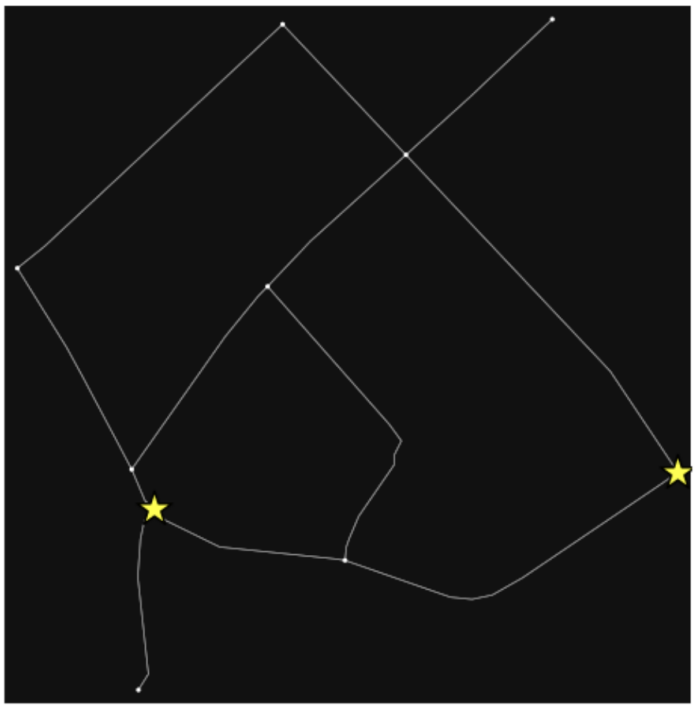

Figure 6. A sample task graph with 2 trucks.

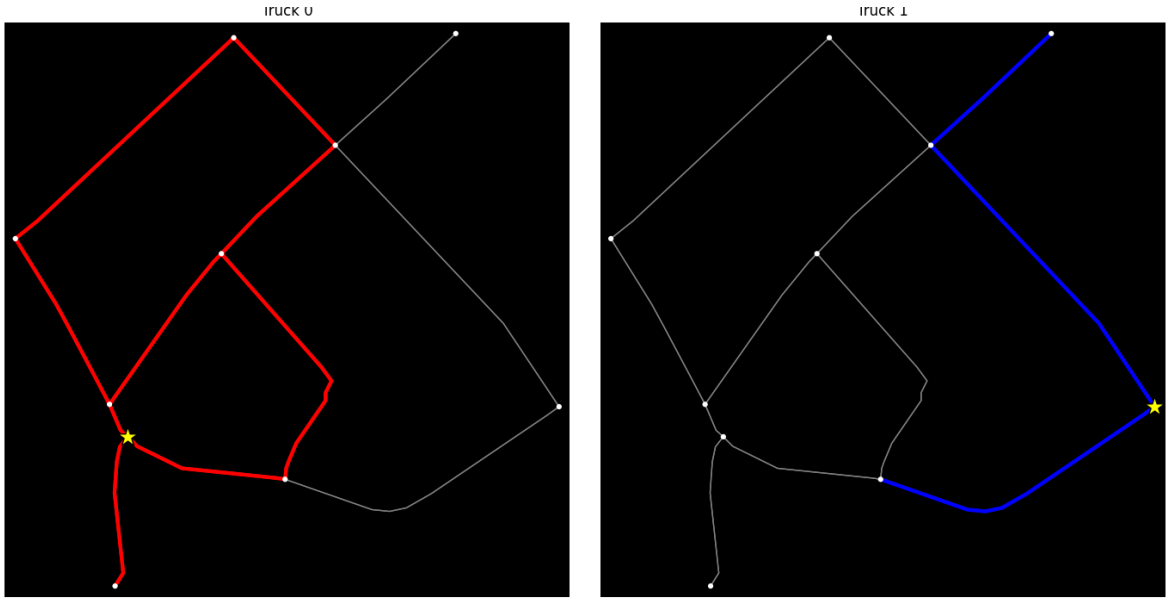

Figure 7. An optimal MILP solution for sample task graph with 2 trucks.

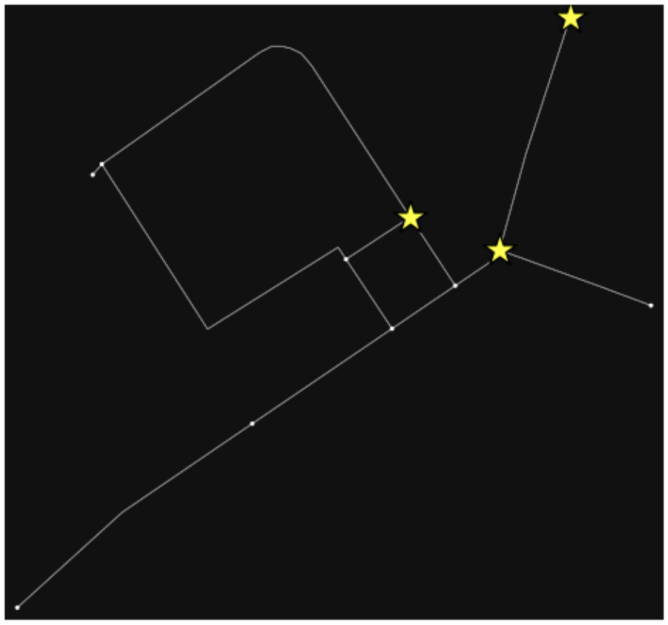

Figure 8. An sample task graph with 3 trucks.

The model also constructs a return subgraph used only for inference to compute the shortest detours back to the depot, as explained in Chapter 3. All tensors are padded and stacked using a custom `collate_fn`, enabling batched training. For training and inference, the dataset is randomly partitioned into an 80% training and 20% validation subset using a fixed PyTorch random generator with seed=42. In the forward pass, the model computes edge-assignment logits on the line graph and pointer-sequence logits for every mini-batch under teacher forcing. The composite loss is computed, and during backpropagation, the gradients are clipped to a norm of 1.0 to prevent explosion in the Transformer. After each epoch, an out-of-sample greedy decode is executed on the validation set, reporting the average normalized edit distance [29] and the token-level accuracy to evaluate the quality of the sequence. The service coverage, defined as the fraction of the tasks required to be traversed by any predicted truck, is also computed and investigated.

5.2.1 RL Routing Implementation

This section documents the implementation choices made to build, train, and evaluate the proposed reinforcement learning (RL) routing architecture. The main road network $G = (V, E)$ used is the same Tartu graph used in previous experiments. The data set is divided into 80% for training and 20% for evaluation of the tests.

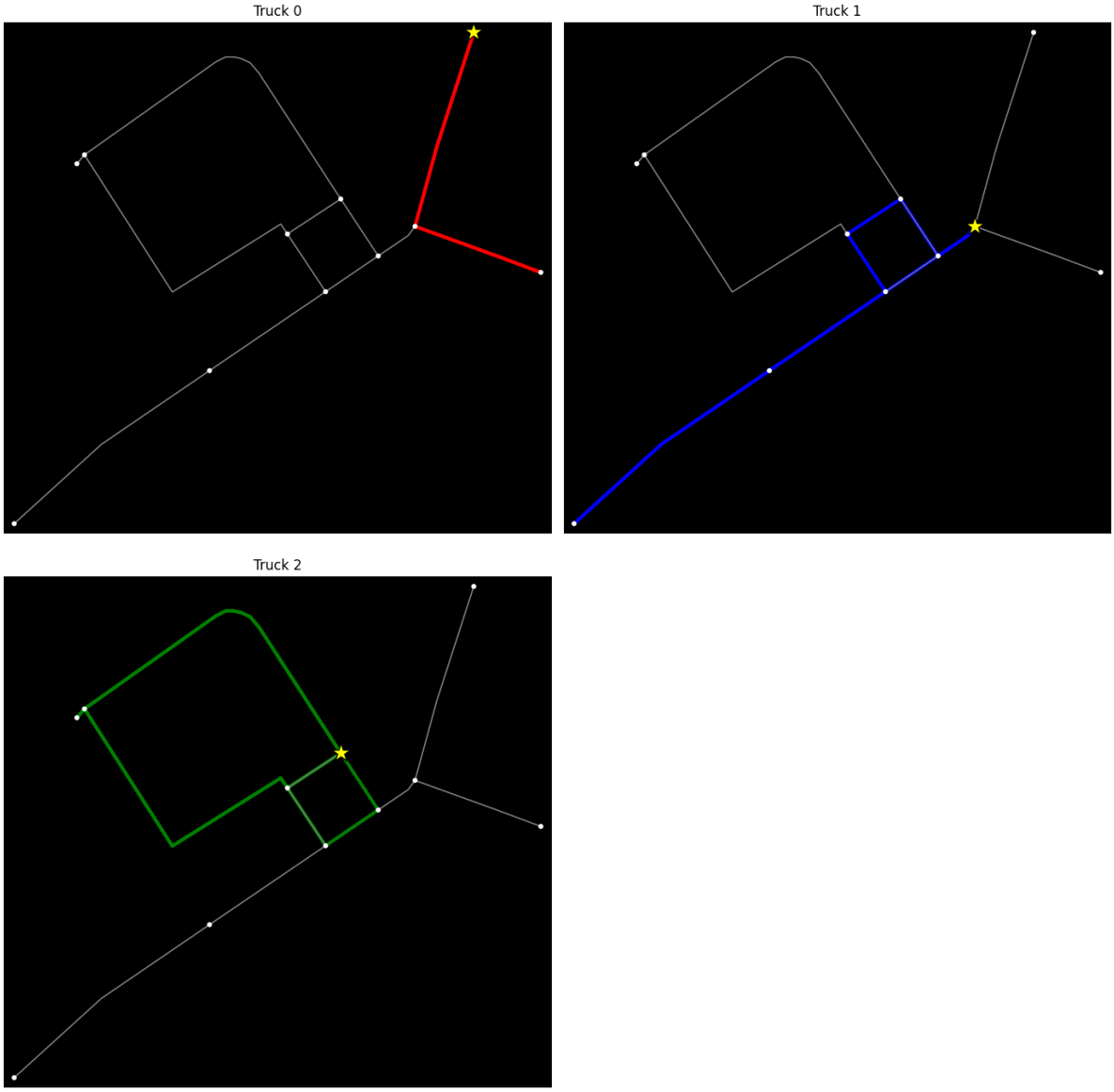

Figure 9. An optimal MILP solution for sample task graph with 3 trucks.

A random seed of 42 is passed to random split so that the split is deterministic to keep training and evaluation data consistent across all models implemented in this research. The actor–critic routine jointly trains the pointer decoder (actor) and the head of state value (critic) on the objective of the policy gradient with Advantage = $R_i - V_\theta(s_i)$ with a fixed coefficient $\lambda_{\text{critic}} = 0.5$. The model is trained for 70 epochs. The scalar reward designed in Chapter 3 is instantiated with $(w_{\text{cov}},\ w_{\text{miss}},\ w_{\text{dh}},\ w_{\text{long}},\ w_{\text{ret}}) = (50,\ 50,\ 1,\ 1,\ 1)$ The quality of the model is then evaluated using the deadhead distance, the percentage of coverage of the task edge, and the coefficient of variation (CV) as an indicator of the balance of workload.

The RL section of the two-stage hybrid ML-RL model follows the same implementation and training setup.

5.2.2 Evaluation Metrics

To comprehensively compare and validate the routing models the following metrics are defined:

- The total distance travelled
- The total deadhead distance
- The deadhead ratio
- Workload Coefficient of Variation (CV)
- Coverage percentage

Total Deadhead Travelled

The total distance traveled metric represents the sum of all distances covered by the entire fleet of snowplow vehicles in a given sample. It includes distance spent actively servicing required road segments as well as any travel in between. The total distance traveled is a primary indicator of routing efficiency and cost, where shorter total routes mean less fuel consumption, less time, and lower operating costs. As such, minimizing the total travel distance directly correlates with faster service completion and reduced resource usage.

Total Deadhead Distance

The total deadhead distance is the aggregate distance that vehicles travel without performing any service. In snowplow routing, this includes traveling from a depot to the start of a route or moving between disjoint service segments without plowing [30]. By reducing deadhead travel, routes become more efficient, allowing roads to be cleared faster and resources to be used more effectively [31].

Deadhead Ratio

The deadhead ratio provides a normalized measure of deadheading relative to the required service. It is defined as the total deadhead distance divided by the total length of required edges, which are the roads that must be plowed. Formally, this ratio is computed as follows:

$$\text{Deadhead Ratio} = \frac{\text{Total Deadhead Distance}}{\text{Total Length of Required Edges}} \tag{14}$$

This dimensionless metric indicates the proportion of non-service travel a solution incurs compared to the essential plowing distance. A lower dead-head ratio means more efficient routing, as vehicles travel relatively little distance without servicing.

Workload Coefficient of Variation (CV)

The coefficient of variation of workload (CV) assesses the balance of the lengths of the routes among all vehicles in a sample fleet. The Workload CV is the standard deviation of individual route distances divided by their mean:

$$\text{Workload CV} = \frac{\sigma(D)}{\mu(D)}, \tag{15}$$

where D is the set of distances each truck travels on its route. This coefficient of variation dispersion [32] indicates how evenly or unevenly the plowing work is distributed. A CV of 0 would mean that all vehicles have equal route lengths, achieving perfect workload balance. Higher CV values imply greater disparity, where some trucks cover significantly longer distances, while others have shorter routes. In a well-balanced snowplow routing solution, the CV workload is kept low, ensuring that no single vehicle or driver is overburdened while others underutilize capacity [33].

Coverage Percentage

The coverage percentage measures the completeness of service indicated by the proportion of required road segments that are plowed by the solution. It is defined as:

$$\text{Coverage} = \frac{\text{Number of Serviced Required Edges}}{\text{Total Number of Required Edges}} \times 100\% \tag{16}$$

giving the fraction of required edges covered, expressed as a percentage. This metric is critical in the context of snowplow routing because the ultimate goal is to service all designated streets that need to be cleared of snow. [31] A coverage of 100% means the solution successfully plowed every required road, while any value less than 100% indicates that some edges were

left unserviced. In practice, route plans must strive for full coverage; missing even a small percentage of required edges can be unacceptable in winter road maintenance, as unplowed roads pose safety hazards.

6. Results and Analysis

This section presents a comparative analysis of the three machine learning approaches evaluated for the snowplow arc routing problem: (i) a purely supervised learning (SL) model, (ii) a reinforcement-learning (RL) model, and (iii) a hybrid two-stage model in which an RL routing procedure follows a supervised edge-assignment network. All models were assessed with the same test set of 114 road network instances generated from the optimal mixed-integer linear programming (MILP) solutions. For each instance, the quality of the solution is evaluated based on the total distance traveled by the trucks, the deadhead distance, and workload balance. Additionally, the pre-processed operational feasibility of the models is also evaluated based on the edge coverage before any heuristic post-processing routine is applied. Table 4 summarizes the mean metric values of the test samples supplied by the experimental pipeline.

Table 4. Summary of model performances on total distance, deadhead distances, deadhead ratio, and workload balance

Metric (mean)	**MILP (Ground-truth)**	**Hybrid ML-RL**	**Reinforcement Learning**	**Supervised**
Total distance (m)	**1570.6**	2897.4	2822.0	3084.6
Total deadhead (m)	**2.1**	2068.1	1866.2	1893.1
Deadhead ratio	**0.002**	1.468	1.530	**1.422**
Maximum truck distance (m)	1116.8	**912.2**	991.0	9231
Coverage before post-proc. (%)	-	**70.31**	65.06	65.64
Workload Coefficient of Variation	2.254	**0.843**	0.945	0.903

6.1 Total Distance Traveled and Deadhead Distances

All learning methods produced tours that are 80 – 95% longer than the MILP optimum ground truth values. The pure reinforcement learning (RL) model achieved the lowest average total distance among the learning approaches at 2822 meters, edging out the supervised learning (SL) model by 2.6 % and the hybrid model by 8.5 %. This low value suggests that the pure RL model produced the least deadhead of the three models. The same ordering performance is observed for the total deadhead distance. In a similar vein, the ratio of deadhead to total required distance of task edges, a metric that arguably more clearly measures the efficiency of the model, shows that the RL model has the best ratio of 1.42, followed by the SL model with a ratio of 1.47, and then

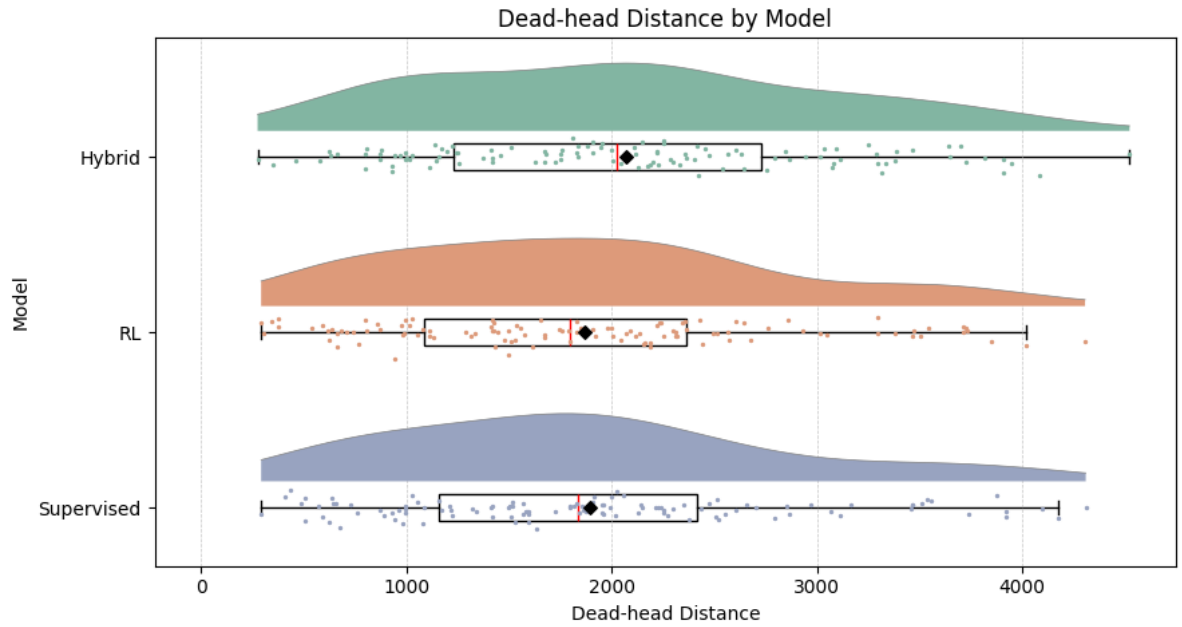

Figure 10. The distribution of the deadhead distances per model over the test samples (Lower is Better).

the hybrid model of 1.53. Figure 10 shows the distribution of the deadhead distance between all 114 test samples for each learning model, and Figure 11 visualizes the deadhead ratios of each for each learning model. It can be seen from the box plots in these visualizations that the RL model achieves the lowest deadhead distance and deadhead ratios, with the distribution skewing more towards the lower distance values.

6.2 Coverage and Workload Balance

Even though the hybrid model records the highest total mileage in the deadhead distance, it also records the lowest maximum per-truck mileage at 912 meters, indicating a relatively balanced work distribution. This apparent contradiction is resolved by inspecting the coefficient of variation. The workload coefficient of variation (CV) results support these observations by capturing the fairness in route-length distribution. The hybrid model achieves the lowest CV of 0.84, a 7% improvement over SL and a 11% improvement over RL. Interestingly, the ground truth optimization model has a CV of 2.25, indicating that strict optimality in distance does not necessarily coincide with equitable workload, a trade-off also observed in the related vehicle routing literature [34]. Figure 12 shows the distribution of the workload balances between the networks in the test sample. It can be observed in this graph that the mean CV value for the test samples is lowest for the hybrid model, while the MILP model has the highest, indicating a disparity in workload equity between the trucks.

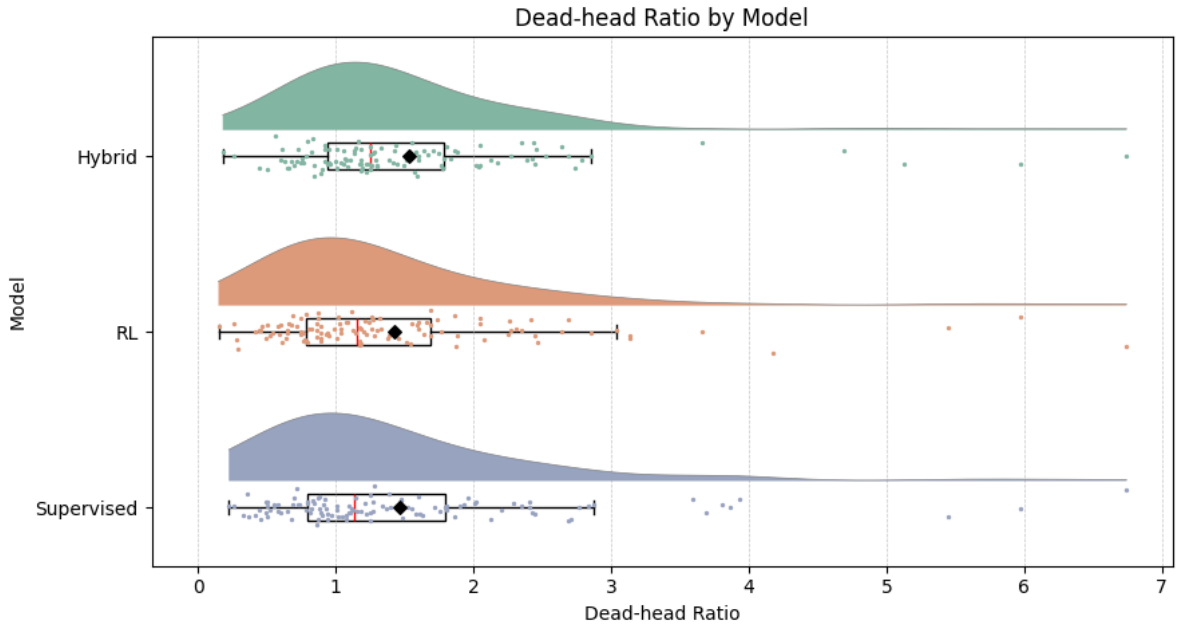

Figure 11. The distribution of the deadhead ratio per model over the test samples (Lower is Better).

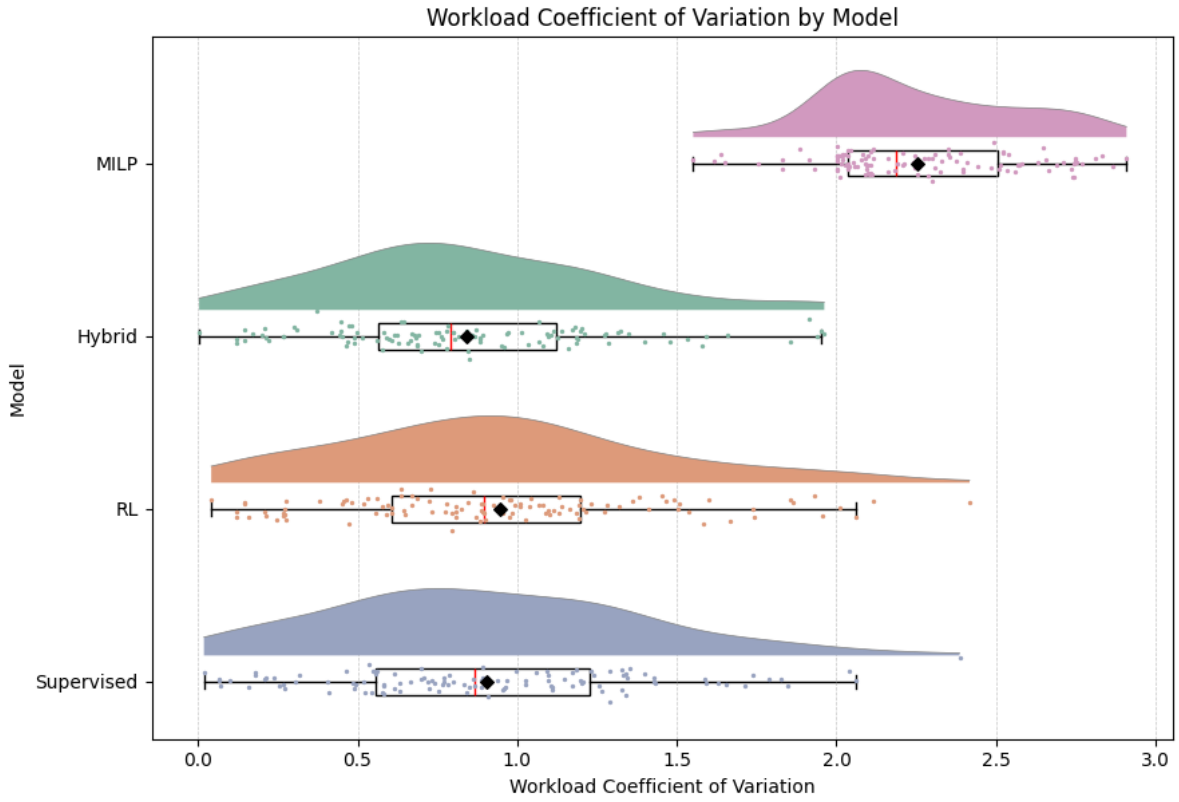

Figure 12. The distribution of the coefficient of variation over the test samples indicating the workload balance on trucks. (Lower is Better).

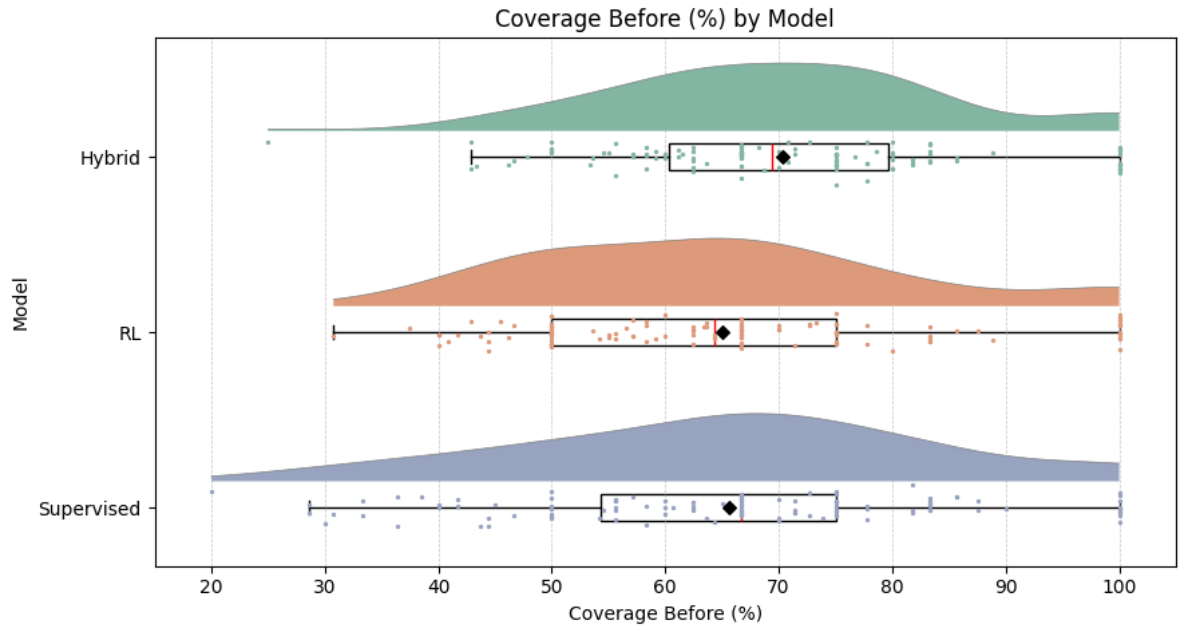

Figure 13. The distribution of task coverage percentage before post-processing (Higher is Better).

The hybrid model also exhibits the best coverage performance among the three learning models with a coverage percentage of 70%, the supervised model follows with a coverage percentage of 65.6% and the RL closely follows with 65%. These results imply that the decoupled supervised edge-assignment network effectively partitions the graph into subtours, mimicking the optimum before the RL stage refines them. However, while the hybrid model gains some advantage from the decoupled preceding supervised assignment stage, its reward function may be overcompensating for achieving workload balancing, favoring balanced tours more than reduced deadhead. The better performance of the hybrid model on task coverage before can be seen in Figure 13, showing the mean coverage percentage of the distribution of each model's coverage percentages across the test sample.

Figure 14 also presents a sample test road network showing each of the model's coverage before preprocessing. As observed, the hybrid model has the best coverage.

In Figure 15 the truck routes for the sample after post-processing are shown. The hybrid model has the highest deadhead ratio, a consequence of trying to balance workload.

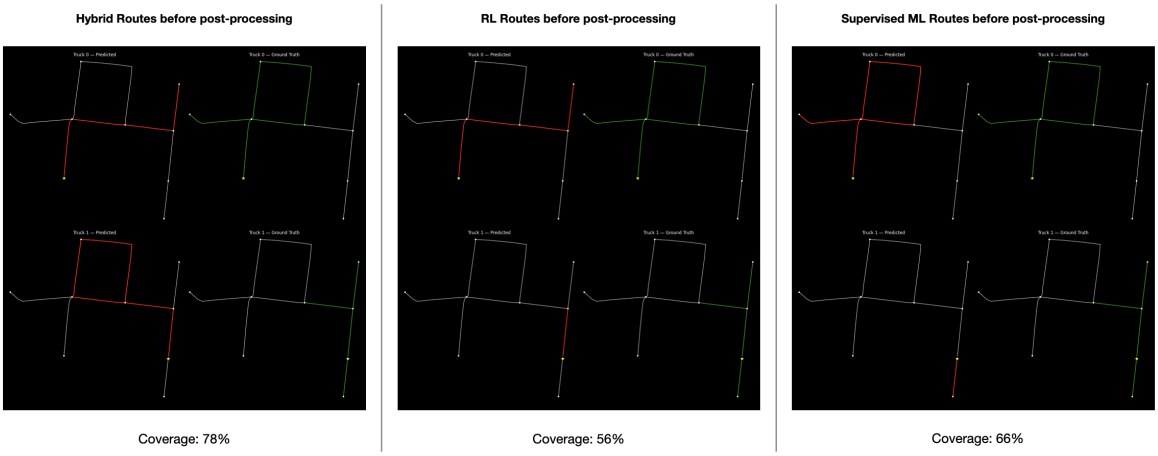

Figure 14. Side by side comparison of the coverage percentage of the models before post-processing on one sample road network

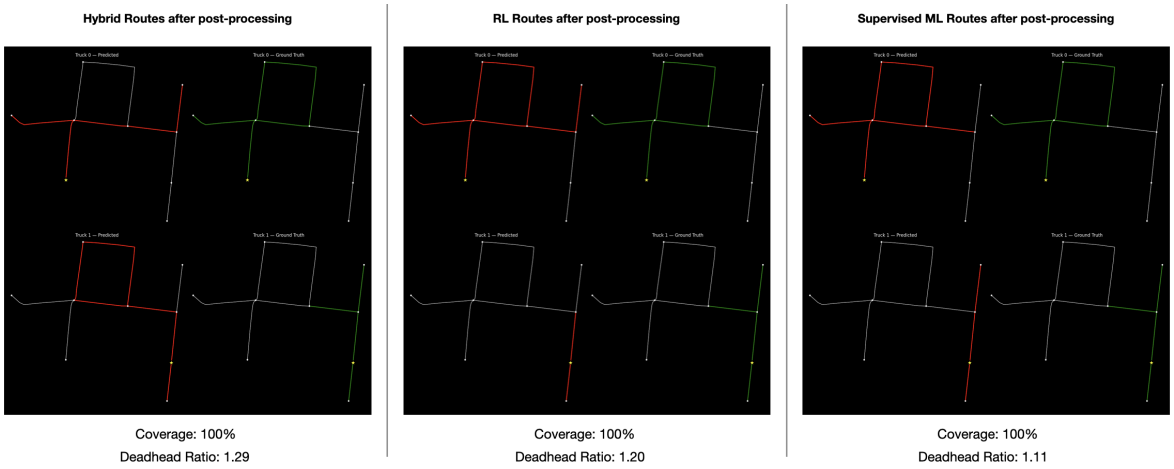

Figure 15. Side by side comparison of the coverage percentage of the models after post-processing on one sample road network

7. Discussion

The empirical evidence from this research yields three principal insights. Firstly, reinforcement learning (RL) remains the most distance-efficient learning paradigm, suggesting that directly learning a sequential decision policy yields lower total and dead-head distances. This aligns with previous findings that policy-gradient methods can internalize complex global objectives with sufficient exploration [8]. Secondly, task decomposition enhances workload balance, but at a cost. The hybrid's superior workload coefficient of variation demonstrates that there is value in decoupling assignment from routing, allowing for each model to build expertise in a specific task. However, the elevated dead-head ratio implies that misassignments can potentially negate a portion of the RL decoder's optimization capacity, but these could be assuaged with a comprehensive reward structure. Finally, heuristic post-processing remains indispensable. Edge-coverage deficits exceeding 30% observed before repair highlight limitations of current decoder architectures in adhering to hard-coverage constraints. Although post-processing can recover any residual task edges not covered, coverage-aware training objectives or constrained decoding strategies could help remove the need for any extra post-processing. In general, the RL model is the most promising foundation for operational deployment when prioritizing distance efficiency. However, the hybrid architecture offers advantages in equitable workload distribution, which is an important practical consideration for snowplow truck routing, maintenance, and planning. Bridging the two objectives remains an open avenue, potentially addressable through multi-objective RL or reinforcement-constrained optimization frameworks.

8. Conclusion and Future Work

8.1 Conclusion

This thesis research addresses the critical problem of optimizing snowplowing operations on Estonian roads by tackling the Snowplow Arc Routing Problem (SARP). The research was driven by four critical research questions defined in Chapter 1. To address the use of Mixed Integer Linear Programming (MILP) to solve the Snowplow Arc Routing Problem (SARP), an MILP model capable of producing exact and optimal solutions was formulated and implemented. The MILP solutions provided critical benchmarks and reliable ground-truth data but exhibited significant computational limitations at scale, emphasizing their practicality primarily for smaller scenarios. A supervised Graph Neural Network (GNN) with Transformer-based pointer decoders was implemented. This supervised model successfully approximated optimal MILP solutions, achieving substantial operational feasibility. However, the model frequently required heuristic post-processing due to incomplete edge coverage before correction, highlighting challenges in enforcing strict connectivity and coverage constraints. A reinforcement learning (RL) framework was also developed that utilized an actor-critic model. This framework enabled dynamic route optimization with improved distance efficiency compared to the purely supervised approach. A hybrid ML-RL architecture was also implemented, leveraging supervised learning for edge assignment and RL for routing. This hybrid method notably enhanced coverage feasibility and workload balancing, indicating RL's potential in complementing supervised ML to address inherent connectivity and coverage limitations. Through rigorous experimentation with real-world data from Estonian Tartu road networks, the research confirms the practicality of ML and RL techniques for scalable routing solutions, highlighting important trade-offs between route optimality, workload balance, and coverage completeness.

8.2 Future Work

The findings presented in this thesis open promising avenues for future research, each with significant potential to advance the state-of-the-art in arc routing optimization. Future works could explore more sophisticated reward structures that incorporate multi-objective optimization methods. Specifically, designing rewards that explicitly balance deadhead reduction, workload equity, and complete task coverage could enhance solution quality and operational efficiency. In addition, extending the current RL framework to explicitly handle real-time disruptions such as traffic variability and equipment failures would considerably improve the operational robustness of snowplow routing and could be an excellent opportunity for further research. In addition,

investigating methods that improve the generalizability of ML-based models across substantially different network topologies or varying operational conditions is another essential area.

In conclusion, the results of this thesis provide significant practical solutions for Estonian road maintenance in winter conditions and establish foundational knowledge and methodologies that are broadly applicable to complex routing problems. The continued exploration of these proposed research directions promises advancements in efficient, adaptive, and robust routing systems essential for modern infrastructure management.

References

[1] Dror M. Arc routing: theory, solutions and applications. Springer Science & Business Media, 2012.

[2] Hess C., Dragomir A. G., Doerner K. F., and Vigo D. Waste collection routing: a survey on problems and methods. *Central European Journal of Operations Research* 32.2 (2024), pp. 399–434.

[3] Laporte G. Fifty years of vehicle routing. *Transportation science* 43.4 (2009), pp. 408–416.

[4] Hajibabai L., Nourbakhsh S. M., Ouyang Y., and Peng F. Network routing of snowplow trucks with resource replenishment and plowing priorities: Formulation, algorithm, and application. *Transportation Research Record* 2440.1 (2014), pp. 16–25.

[5] Corberán Á. and Laporte G. Arc routing: problems, methods, and applications. SIAM, 2015.

[6] Corberán A., Eglese R., Hasle G., Plana I., and Sanchis J. M. Arc routing problems: A review of the past, present, and future. *Networks* 77.1 (2021), pp. 88–115.

[7] Perrier N., Langevin A., and Campbell J. F. A survey of models and algorithms for winter road maintenance. Part I: system design for spreading and plowing. *Computers & Operations Research* 33.1 (2006), pp. 209–238.

[8] Nazari M., Oroojlooy A., Snyder L., and Takác M. Reinforcement learning for solving the vehicle routing problem. *Advances in neural information processing systems* 31 (2018).

[9] Cabral E. A., Gendreau M., Ghiani G., and Laporte G. Solving the hierarchical Chinese postman problem as a rural postman problem. *European Journal of Operational Research* 155.1 (2004), pp. 44–50.

[10] Arakaki R. K. and Usberti F. L. Hybrid genetic algorithm for the open capacitated arc routing problem. *Computers & Operations Research* 90 (2018), pp. 221–231.

[11] Festa P. A brief introduction to exact, approximation, and heuristic algorithms for solving hard combinatorial optimization problems. *2014 16th International Conference on Transparent Optical Networks (ICTON)*. IEEE. 2014, pp. 1–20.

[12] Clarke G. and Wright J. W. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research* 12.4 (1964), pp. 568–581.

[13] Glover F., Kelly J. P., and Laguna M. Genetic algorithms and tabu search: hybrids for optimization. *Computers & Operations Research* 22.1 (1995), pp. 111–134.

[14] Kool W., Van Hoof H., and Welling M. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475* (2018).

[15] Kool W., Hoof H. van, Gromicho J., and Welling M. Deep policy dynamic programming for vehicle routing problems. *International conference on integration of constraint programming, artificial intelligence, and operations research*. Springer. 2022, pp. 190–213.

[16] Vinyals O., Fortunato M., and Jaitly N. Pointer networks. *Advances in neural information processing systems* 28 (2015).

[17] Larsen J. Vehicle routing with time windows—finding optimal solutions efficiently. 1999.

[18] Sutton R. S., Barto A. G., et al. Reinforcement learning: An introduction. Vol. 1. 1. MIT press Cambridge, 1998.

[19] Silver D., Huang A., Maddison C. J., Guez A., Sifre L., Van Den Driessche G., Schrittwieser J., Antonoglou I., Panneershelvam V., Lanctot M., et al. Mastering the game of Go with deep neural networks and tree search. *nature* 529.7587 (2016), pp. 484–489.

[20] Haklay M. and Weber P. Openstreetmap: User-generated street maps. *IEEE Pervasive computing* 7.4 (2008), pp. 12–18.

[21] Boeing G. OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, environment and urban systems* 65 (2017), pp. 126–139.

[22] Gurobi Optimizer Reference Manual. https://www.gurobi.com/documentation/current. Gurobi Optimization, LLC. 2023.

[23] Tirkolaee E. B., Alinaghian M., Hosseinabadi A. A. R., Sasi M. B., and Sangaiah A. K. An improved ant colony optimization for the multi-trip Capacitated Arc Routing Problem. *Computers & Electrical Engineering* 77 (2019), pp. 457–470.

[24] Kipf T. N. and Welling M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[25] Soffer S. N. and Vazquez A. Network clustering coefficient without degree-correlation biases. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* 71.5 (2005), p. 057101.

[26] Zhang J. and Luo Y. Degree centrality, betweenness centrality, and closeness centrality in social network. *2017 2nd international conference on modelling, simulation and applied mathematics (MSAM2017)*. Atlantis press. 2017, pp. 300–303.

[27] Simonovsky M. and Komodakis N. Dynamic edge-conditioned filters in convolutional neural networks on graphs. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 3693–3702.

[28] Grondman I., Busoniu L., Lopes G. A., and Babuska R. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, part C (applications and reviews)* 42.6 (2012), pp. 1291–1307.

[29] Marzal A. and Vidal E. Computation of normalized edit distance and applications. *IEEE transactions on pattern analysis and machine intelligence* 15.9 (2002), pp. 926–932.

[30] Nguyen P. H. and Tran D. Optimizing snow and ice route removal operations using vehicle routing problems and geographic information system. *Journal of Cold Regions Engineering* 38.2 (2024), p. 04024001.

[31] Zhang D. Optimization of vehicle routing for plowing and snow disposal. *ICCTP 2009: Critical Issues In Transportation Systems Planning, Development, and Management*. 2009, pp. 1–7.

[32] Abdi H. Coefficient of variation. *Encyclopedia of research design* 1.5 (2010), pp. 169–171.

[33] Lacomme P., Prodhon C., Prins C., Gandibleux X., Beillevaire B., and Ren L. A split based approach for the vehicle routing problem with route balancing. *International Conference on Operations Research and Enterprise Systems*. Vol. 2. SCITEPRESS. 2014, pp. 159–166.

[34] Matl P., Hartl R. F., and Vidal T. Workload equity in vehicle routing problems: A survey and analysis. *Transportation Science* 52.2 (2018), pp. 239–260.

Appendix

GitHub Repository Link

The implementation and experiment code for this research can be found here.

License

Non-exclusive licence to reproduce the thesis and make the thesis public

I, **Obed Kobina Nsiah**

1. grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the digital archives of the University of Tartu until the expiry of the term of copyright, my thesis **Optimized Snowplow Routing on Estonian Roads: Machine Learning Approaches**, supervised by Dr. Kallol Roy and Dr. Jaan Übi;

2. grant the University of Tartu a permit to make the thesis specified in point 1 available to the public via the web environment of the University of Tartu, including via the digital archives, under the Creative Commons licence CC BY NC ND 4.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright;

3. am aware of the fact that the author retains the rights specified in points 1 and 2;

4. confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Obed Kobina Nsiah

15/05/2025