EasyChair Preprint

№ 14918

EFECTIW-ROTER: Deep Reinforcement Learning Approach for Solving Heterogeneous Fleet and Demand Vehicle Routing Problem With Time-Window Constraints

Arash Mozhdehi, Mahdi Mohammadizadeh, Yunli Wang, Sun Sun and Xin Wang

September 17, 2024

EFECTIW-ROTER: Deep Reinforcement Learning Approach for Solving Heterogeneous Fleet and Demand Vehicle Routing Problem With Time-Window Constraints

Arash Mozhdehi
arash.mozhdehi@ucalgary.ca
University of Calgary
Calgary, Alberta, Canada

Mahdi Mohammadizadeh
mahdi.mohammadizadeh@ucalgary.ca
University of Calgary
Calgary, Alberta, Canada

Yunli Wang
yunli.wang@nrc-cnrc.gc.ca
National Research Council
Ottawa, Ontario, Canada

Sun Sun
sun.sun@nrc-cnrc.gc.ca
National Research Council
Waterloo, Ontario, Canada

Xin Wang*
xcwang@ucalgary.ca
University of Calgary
Calgary, Alberta, Canada

ABSTRACT

The heterogeneous fleet and demand vehicle routing problem with time-window constraints (HFDVRPTW) is a crucial optimization problem of significant importance in real-world logistics operations. In this paper, we propose a deep reinforcement learning (DRL)-based method, termed spatial Edge-Feature EnhanCed mulTI-graph fusion encoder With spectral-based embedding and hieRarchical decOder with learnable TEmpoRal positional embedding (EFECTIW-ROTER, pronounced "Effective Router"), to tackle this complex and practical optimization problem. EFECTIW-ROTER utilizes two sparse graphs to represent node connectivity, where nodes correspond to customers and the depot. This sparsity results from the time-window constraints and customers' demand relative to the list of acceptable vehicle attributes specified for service within a heterogeneous fleet, determined by the reachability of the nodes based on these two factors. Leveraging two graph Transformer models, EFECTIW-ROTER's encoding module captures the interactions between the nodes based on these factors. One model encodes customers' heterogeneous demand with spatial edge features based on travel time between the nodes, while the second employs temporal positional embeddings to capture temporal relationships based on time-window ordering. A fusion model is introduced to integrate node interactions based on these graphs. Additionally, a spectral-attention-based pooling ensures effective state representation for the DRL-based method. EFECTIW-ROTER features a hierarchical attention decoder operating in two stages: heterogeneous vehicle selection and node selection. Enhanced with positional embeddings, the decoder is empowered to make effective routing decisions based on time-window constraints' ordering. Experimental results using real-world traffic data from two major Canadian cities confirm EFECTIW-ROTER's better performance over current state-of-the-art DRL-based and heuristic methods. EFECTIW-ROTER reduces travel times while also achieving faster computational times when compared to conventional heuristics. Additional experiments demonstrate its generalizability across larger instances.

CCS CONCEPTS

• **Mathematics of computing → Combinatorial optimization**; • **Information systems → Spatial-temporal systems**; • **Computing methodologies → Sequential decision making**; • **Theory of computation → Reinforcement learning**;

KEYWORDS

Combinatorial Optimization, Reinforcement Learning, Attention Model, Spatial-Temporal systems

ACM Reference Format:
Arash Mozhdehi, Mahdi Mohammadizadeh, Yunli Wang, Sun Sun, and Xin Wang. 2024. EFECTIW-ROTER: Deep Reinforcement Learning Approach for Solving Heterogeneous Fleet and Demand Vehicle Routing Problem With Time-Window Constraints. In *The 32nd ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL '24), October 29-November 1, 2024, Atlanta, GA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3678717.3691208

1 INTRODUCTION

The heterogeneous fleet and demand vehicle routing problem with time-window constraints (HFDVRPTW) is a critical problem in optimizing real-world delivery operations. Customer constraints on vehicle capacity for real-world deliveries are influenced by factors such as unloading dock dimensions and road network limitations [14]. In addition, the need for specialized vehicles, such as refrigerated ones for perishable cargo, underscores the importance of considering both the fleet and demand heterogeneity in optimizing real-world logistics operations. This variant of the vehicle routing problem (VRP) allows customers to choose a list of combinations of vehicle capacity and type from all available options to best meet their delivery needs. It also incorporates customer-specified time-window constraints as a result of urban traffic regulations and customers' operational rules. The HFDVRPTW is an optimization

*Corresponding author

SIGSPATIAL '24, October 29-November 1, 2024, Atlanta, GA, USA

ACM ISBN 979-8-4007-1107-7/24/10
https://doi.org/10.1145/3678717.3691208

problem seeking to minimize total operational costs by determining optimal vehicle routes to serve the demand of geographically dispersed customers within designated time windows, ensuring that the vehicle type and capacity match one of the options specified by each customer.

VRPs, in general, are NP-hard combinatorial optimization problems. Exact methods, such as branch-and-bound and branch-price-and-cut, fail to reach solutions within a reasonable time for large-scale, real-world problems due to their worst-case exponential computational complexity. Heuristic algorithms, e.g. sweep algorithm and ant colony optimization (ACO), on the other hand, trade the guarantee of finding an optimal solution in favor of speed [18]. However, the effectiveness of these methods significantly relies on human expertise, necessitating cumbersome processes of hand-crafting the search rule. Moreover, these methods lack flexibility and generalizability, as outlined by *No Free Lunch Theorem* [1, 34]. Recent studies, driven by advances in utilizing deep reinforcement learning (DRL) techniques for various combinatorial optimization problems, have leveraged this progress to tackle VRPs. In particular, DRL-based methods with Transformer-style policy networks demonstrated promising results in solving this class of optimization problems [23, 35]. However, these methods have the following shortcomings that would make them ineffective for solving HFD-VRPTW:

(1) **Inability to leverage the graph connectivity inductive bias resulting from time-windows and demand heterogeneity:** When considering the heterogeneity of customer demand, it is clear that nodes representing customers with different vehicle attribute preferences, i.e., capacity or type, cannot be reached from each other. Similarly, when the time-windows of two customer nodes do not overlap, considering the travel time between them, these nodes are unreachable directly from each other (due to hard time-windows). Effectively capturing the relationships between the nodes under these two considerations requires accounting for the graph sparsity that results from these factors. However, existing DRL-based methods utilize the original Transformer architecture designed for natural language processing, which necessitates a connection between all nodes, as is the case with words in a sequence. The original Transformer lacks inductive biases required to capture relationships between nodes according to graph topology, leading to poor performance in graph-based tasks [6, 21]. This shortfall renders current DRL-based methods ineffective in routing when considering these two factors, as effective routing is entangled with the model's performance in capturing node relationships [32].
(2) **Lack of state representation aligned with time-windows and demand heterogeneity:** Effective state representation is vital for successful policy search in DRL-based approaches [22, 36]. While current methods utilize graph embeddings to reflect the structure of problem instances in state representation for effective policy search [35], their employed graph embedding methods fall short in adequately capturing the underlying graph structure due to their inherently 'flat' graph summarization process [8, 37]. This limitation is more pronounced for HFDVRPTW, where capturing local neighborhood structures based on graph topology is essential due to sparse nature of the graph (as explained in the previous paragraph). In fact, the sparsity exacerbates the inadequacies of graph-level representation techniques used in existing methods, as they neglect the structure of the graph [37], making them ineffective at capturing the essential structural nuances shaped by time-windows and demand heterogeneity, which is vital for effective state representation.
(3) **Inability to exploit the time-window-based ordering between the neighboring nodes:** According to studies in the operations research (OR) literature, considering time-window constraints in VRPs, the relative temporal ordering of time-windows designated by customers among customer nodes can provide important insights for routing by guiding the solution search [7, 9, 11, 25]. However, existing DRL-based approaches using Transformer-based architectures fail to consider this ordering, assuming no inherent ordering among nodes [15]. Moreover, unlike recurrent neural networks (RNNs), Transformers do not inherently model any temporal order of input within their architecture [29, 31]. This failure to maintain ordering according to time-window constraints results in the inability to capture temporal relationships between customers, which can render these methods ineffective for routing with time-window constraints.

To solve HFDVRPTW, in this paper, we propose a DRL-based method, named spatial Edge-Feature EnhanCed mulTI-graph fusion encoder With spectral-based embedding and hieRarchical decOder with learnable TEmpoRal positional embedding (EFECTIW-ROTER pronounced "Effective Router"). The major contributions of this paper are summarized as follows:

- We employ two distinct sparse graphs to model the relationships between nodes, shaped by time-window constraints and demand heterogeneity. We propose utilizing graph Transformer models to leverage the connectivity inductive bias for encoding these graphs. A fusion model is introduced to integrate node interactions, capturing the underlying structure of the routing problem instance considering these factors. To enhance routing performance, our graph Transformer model encodes heterogeneous demand by incorporating spatial features derived from road network travel times between locations as edge features in the encoding process.
- We propose utilizing a spectral-attention-based graph pooling technique to enhance the graph-level embedding. This approach addresses the lack of state representation aligned with time-windows and demand heterogeneity by effectively capturing underlying graph structure based on two factors.
- We introduce a hierarchical attention decoder designed to perform route construction for a heterogeneous fleet, ensuring the assignment of appropriate vehicle types to meet customer demands. The decoder operates in two stages: initially, it selects a vehicle from the heterogeneous fleet; subsequently, it determines next node for the vehicle to visit.
- We propose the incorporation of learnable temporal positional embeddings in both encoding and decoding processes. By integrating these embeddings, the encoder can capture

the temporal relationships between neighboring nodes, reflecting their time-window constraint-based ordering. The decoder leverages these temporal positional embeddings in node selection, enabling the policy model to make effective routing decisions based on the time-window order of neighboring customers, tailored to the current state of the vehicle.
- Through experiments performed on problem instances derived from the traffic data of two major cities in Canada, Calgary and Edmonton, we demonstrated the better routing performance of EFECTIW-ROTER for HFDVRPTW compared to both state-of-the-art DRL-based and heuristic-based methods, particularly in terms of total travel time. Additionally, our results show that EFECTIW-ROTER achieves faster computational times than heuristic methods. We demonstrate the generalizability of EFECTIW-ROTER in solving larger-sized problem instances.

The paper is structured as follows: Section 2 provides a brief overview of related works, Section 3 defines HFDVRPTW formally, Section 4 details the EFECTIW-ROTER framework, Section 5 presents experimental results, and Section 6 discuss the future research direction.

2 RELATED WORKS

In this section, we briefly review existing work proposing DRL-based approaches for solving VRPs.

The study by Nazari et al. [24] represents one of the initial efforts to take advantage of advances in DRL for combinatorial optimization. The authors proposed an algorithm that incorporates an RNN-based decoder with an attention mechanism, specifically designed to address the basic VRP. The training utilized an aggregated Euclidean-based measure of travel distance as the feedback signal. Kool et al. [15] adapted a Transformer-based policy network trained using a self-critic reinforcement learning algorithm, which successfully outperformed established baselines, including Google OR-Tools. Duan et al. [4] underscored the ineffectiveness of Kool et al.'s method when the objective is to minimize the total actual travel distance instead of the Euclidean one. Duan et al. highlighted the limitations of Kool et al.'s method, particularly its ineffectiveness in minimizing the total actual travel distance as opposed to the Euclidean distance, through their experiments. Alternatively, they introduced a policy network comprising a graph convolutional network (GCN)-based encoder and two distinct decoders: one employing gated recurrent units (GRU) and the other relying on a multi-layer perceptron (MLP), which successfully outperformed Google OR-Tools in solving the VRP with the objective of minimizing the total road network distance. However, the GRU-based decoder adopted by this DRL-based approach lacked the parallelism and computational efficiency exhibited by the attention-based decoder utilized by Kool et al. Lei et al. [18] disregarded the widely recognized ineffectiveness [16, 17] of estimating road network travel distances using Euclidean distances, instead focusing on solving the heterogeneous-sized fleet VRP. To account for fleet heterogeneity in terms of capacity, the authors extended Kool et al.'s policy model and introduced an MLP-based decoder for vehicle selection. However, they overlooked that vehicle heterogeneity often arises from customer preferences for specific vehicle types. These preferences are influenced by factors such as the dimensions of unloading docks and limitations within the road network. Consequently, different customers may require different types of vehicles, tailored to their unique logistical needs and constraints. Moreover, all these existing methods employ a 'flat' graph summarization process and rely on an ineffective problem instance representation in the state representation.

3 PRELIMINARIES

In this section, we present the preliminaries for HFDVRPTW and introduce an MDP-based formulation for this routing problem.

Definition 3.1. **(Vehicles Feature Vector).** A vehicles feature vector is defined as $\mathcal{F} = \{f_1, \ldots, f_\gamma\}$, where γ denotes the total number of vehicles in a fleet. Each element f_k of this vector is an attribute tuple $(\mathcal{T}_k, Q_k)$, where $\mathcal{T}_k$ and Q_k are the type of vehicle (e.g. refrigerated or non-refrigerated) and capacity of the corresponding vehicle, respectively.

In this paper, we introduce two graph-based representations to model interactions among nodes: a demand graph and a time-window graph. The demand graph represents the spatial relationships between nodes, taking into account their reachability influenced by heterogeneous demand. Conversely, the time-window graph models the temporal interactions among nodes, constrained by their respective time-windows.

Definition 3.2. **(Demand Graph).** A demand graph is a directed graph defined as $G^D = \left(V^D, E^D\right)$, where $V^D = \left\{v_0^D, \ldots, v_C^D\right\}$ represents the set of nodes in the graph. This set include a depot node v_0^D and C customer nodes. To each node v_i^D, a feature vector $\chi_{v_i}^D = (x_i, y_i, d_i)$ and an individual attribute τ_i is ascribed, where x_i and y_i represent the node's 2-dimensional coordinates, with $d_0 = 0$ to indicate no demand for the depot. τ_i represents a list of vehicle attribute tuples, each specifying a vehicle type and capacity that is acceptable to the corresponding customer for fulfilling their demand. The set $V_c^D = V^D \setminus \left\{v_0^D\right\}$ identifies the set of customer nodes. The demand graph's edge set $E^D = \left\{e_{ij}^D | 0 \leq i, j \leq C, i \neq j\right\}$, where to each $e_{ij}^D \in E^D$ an attribute $\epsilon_{ij}^D = tt_{ij}$ assigned, where tt_{ij} denotes the travel time from node v_i^D to node v_j^D. An adjacency matrix $\mathcal{A}^D \in \mathbb{R}^{(C+1)\times(C+1)}$ is defined such that each element a_{ij}^D is equal to 1 if and only if $\tau_i \cap \tau_j \neq \emptyset$ (indicating the existence of an edge between the corresponding nodes), and 0 otherwise (indicating that the corresponding nodes are not connected).

Definition 3.3. **(Time-Window Graph).** A time-window graph is defined as a directed graph $G^{TW} = \left(V^{TW}, E^{TW}\right)$. Each element v_i^{TW} in the node list $V^{TW} = \left\{v_0^{TW}, \ldots, v_C^{TW}\right\}$, which corresponds to the same depot/customer as v_i^D, is assigned a tuple $\chi_{v_i}^{TW} = \left(tw_i^1, tw_i^2\right)$ representing the earliest and latest permissible delivery times. For the depot, specifically v_0^{TW}, the values of tw_0^1 and tw_0^2 are set to 0 and T_{max}, respectively, where T_{max} represents the maximum operational hours of the vehicles. The edge set of the time-window graph G^{TW} is defined as $E^{TW} = \left\{e_{ij}^{TW} \mid 0 \leq i, j \leq C, i \neq j\right\}$. To each edge e_{ij}^{TW} a feature $\epsilon_{ij}^{TW} =$

$tw_j^2 - tw_i^2$ assigned. The adjacency matrix for this graph is denoted by $\mathcal{A}^{TW} \in \mathbb{R}^{(C+1)\times(C+1)}$. Given the traveling time from node i to node j, tt_{ij}, an element a_{ij}^{TW} of the adjacency matrix is set to 1 if there exists $x \in \left[tw_i^1, tw_i^2\right]$ such that $x + tt_{ij} \in \left[tw_j^1, tw_j^2\right]$, and it is set to 0 otherwise.

The process of generating demand and time-window graphs is detailed in Appendix A.

In this paper, we solve the HFDVRPTW in a constructive fashion. We formulate this sequential decision-making task as a Markov decision process (MDP), defined by the tuple $\{\mathcal{S}, A, \mathcal{P}, R\}$, where the elements represent the state space, action space, transition function, and reward function, respectively. The following defines each MDP component for HFDVRPTW at step $\mathcal{T}$.

- **State:** At each step $\mathcal{T}$ of the sequential process, the state $s_{\mathcal{T}}$ is defined as $s_{\mathcal{T}} = \left[s_{\mathcal{T}}^{\mathcal{F}}, s_{\mathcal{T}}^{\mathcal{R}}\right] \in \mathcal{S}$, where $s_{\mathcal{T}}^{\mathcal{F}}$ and $s_{\mathcal{T}}^{\mathcal{R}}$ respectively denote the vehicles state and the routing state. The vehicles state is represented as $s_{\mathcal{T}}^{\mathcal{F}} = \left[s_{\mathcal{T}}^{f_1}, \ldots, s_{\mathcal{T}}^{f_\gamma}\right]$, where each $s_{\mathcal{T}}^{f_k} = \left[rc_{\mathcal{T}}^k, \mathcal{V}_{\mathcal{T}}^k\right]$ characterizes the state of vehicle k. Here, $rc_{\mathcal{T}}^k$ represents the remaining capacity, and $\mathcal{V}_{\mathcal{T}}^k$ denotes the current location of vehicle k. The routing state $s_{\mathcal{T}}^{\mathcal{R}}$ is a vector that includes all customer nodes $v_i^D \in V_c^D$ visited up to the decoding state $\mathcal{T}$.
- **Action:** The action at step $\mathcal{T}$, denoted by $a_{\mathcal{T}} \in A$, is represented as a tuple $\left(f_k, v_{\mathcal{T}+1}^k\right)$, where f_k refers to the chosen vehicle k and $v_{\mathcal{T}+1}^k$ is the next node visited by this vehicle at the subsequent time step.
- **Transition:** The system transition to the next state, $s_{\mathcal{T}+1}$, from the current state, $s_{\mathcal{T}}$, in response to the execution of action $a_{\mathcal{T}}$, is modeled by the function $s_{\mathcal{T}+1} = \mathcal{P}(s_{\mathcal{T}}, a_{\mathcal{T}})$. This function $\mathcal{P}$ maps the current state and action to the subsequent state.
- **Reward:** With the objective of minimizing the aggregated traveling time of the vehicles, the one-step reward at each decision step $\mathcal{T}$ is defined by $R(s_{\mathcal{T}}, a_{\mathcal{T}}) = -tt_{ij}$, where tt_{ij} represents the travel time between nodes i and j.

4 METHODOLOGY

In this section, we present our deep reinforcement learning (DRL)-based method for solving the heterogeneous fleet and demand vehicle routing problem with time-windows (HFDVRPTW), named spatial Edge-Feature EnhanCed mulTI-graph fusion encoder With spectral-based embedding and hieRarchical decOder with learnable TEmpoRal positional embedding (EFECTIW-ROTER). As depicted in Figure 1, EFECTIW-ROTER employs a Transformer-style policy network for sequential route construction. This network consists of two primary components: a spatial edge-feature enhanced multi-graph fusion encoding module with spectral-based embedding and a hierarchical decoding module with learnable temporal positional embedding. Detailed descriptions of these modules follow, along with an explanation of the network training process.

4.1 Spatial Edge-Feature Enhanced Multi-Graph Fusion Encoder with Spectral-based Embedding

For a given instance represented by a demand graph G^D and a time-window graph G^{TW}, the encoder module embeds these problem instances into a higher-dimensional space to facilitate feature extraction. This encoding process commences with the initial embedding of the node features and edges features of G^D and G^{TW}, separately. Given the node v_i^D of the demand graph, with the feature vector $\chi_{v_i} = (x_i, y_i, d_i)$ (as denoted in Definition 3.2), its initial embedding $h_{v_i,0}^D$ is computed through a linear projection with trainable parameters $\omega_0^{D,\chi}$ and $b_0^{D,\chi}$, expressed as $h_{v_i,0}^D = \omega_0^{D,\chi}\chi_{v_i}^D + b_0^{D,\chi}$. For the edge e_{ij} between nodes v_i and v_j in the demand graph, the initial embedding is calculated with learnable parameters $\omega_0^{D,\epsilon}$ and $b_0^{D,\epsilon}$, where $\epsilon_{e_{ij}}^D$ is the edge feature defined in Definition 3.2, given by $h_{e_{ij},0}^D = \omega_0^{D,\epsilon}\epsilon_{e_{ij}}^D + b_0^{D,\epsilon}$. Similarly, for the time-window graph G^{TW}, the initial embeddings for each node v_i^{TW} and edge e_{ij}^{TW} are computed as $h_{v_i,0}^{TW} = \omega_0^{TW,\chi}\chi_{v_i}^{TW} + b_0^{TW,\chi}$ and $h_{e_{ij},0}^{TW} = \omega_0^{TW,\epsilon}\epsilon_{e_{ij}}^{TW} + b_0^{TW,\epsilon}$, respectively. Here, $\chi_{v_i}^{TW}$ and $\epsilon_{e_{ij}}^{TW}$ denote the node feature vector and edge feature for the corresponding node v_i^{TW} and edge e_{ij}^{TW} (as defined in Definition 3.3), respectively. The trainable parameters $\omega_0^{TW,\chi}$, $b_0^{TW,\chi}$, $\omega_0^{TW,\epsilon}$, and $b_0^{TW,\epsilon}$ are used to linear projection of these features.

Following the initial embedding process, the node and edge features of both the demand and time-window graphs undergo transformations via L and L' layers of attention, respectively. The attention mechanism for the demand graph, enhanced with travel time between the nodes as a spatial edge feature, is termed the spatial feature-enhanced demand graph embedder. In contrast, the time-window graph utilizes the time-window graph embedder with temporal positional encoding. Subsequently, the transformed node features from both graphs are integrated using a multi-graph fusion module. A graph embedding is then computed through a spectral-attention-based graph pooling module. Each module is detailed in the following sections.

4.1.1 *Spatial Feature-Enhanced Demand Graph Embedder.*

In the l-th attention layer, the embedding operation begins by performing linear projections on the input node embeddings of the demand graph, $h_{v_i,l}^D$. The *key*, *query*, and *value* for the m-th attention head $\left(m \in \left\{1, \ldots, \mathcal{M}^D\right\}\right)$ are respectively computed as $q_{v_i,l}^m = \omega_{Q,l}^m h_{v_i,l}^D$, $k_{v_i,l}^m = \omega_{K,l}^m h_{v_i,l}^D$, $v_{v_i,l}^m = \omega_{V,l}^m h_{v_i,l}^D$, using trainable parameters $\omega_{Q,l}^m$, $\omega_{K,l}^m$, and $\omega_{V,l}^m$. For each edge feature $h_{e_{ij},l}^D$, a linear transformation is applied via learnable parameters $\omega_{E,l}^m$ to compute the updated edge feature as $\varepsilon_{e_{ij},l}^m = \omega_{E,l}^m h_{e_{ij},l}^D$. Subsequently, using the adjacency matrix entry a_{ij}^D, the attention weight $\alpha_{ij,l}^{D,m}$ for the l-th layer and m-th attention head is calculated as follows:

$$u_{ij,l}^{D,m} = \frac{q_{v_i,l}^{m\,\top} k_{v_i,l}^m}{\sqrt{d_k}}, \tag{1}$$

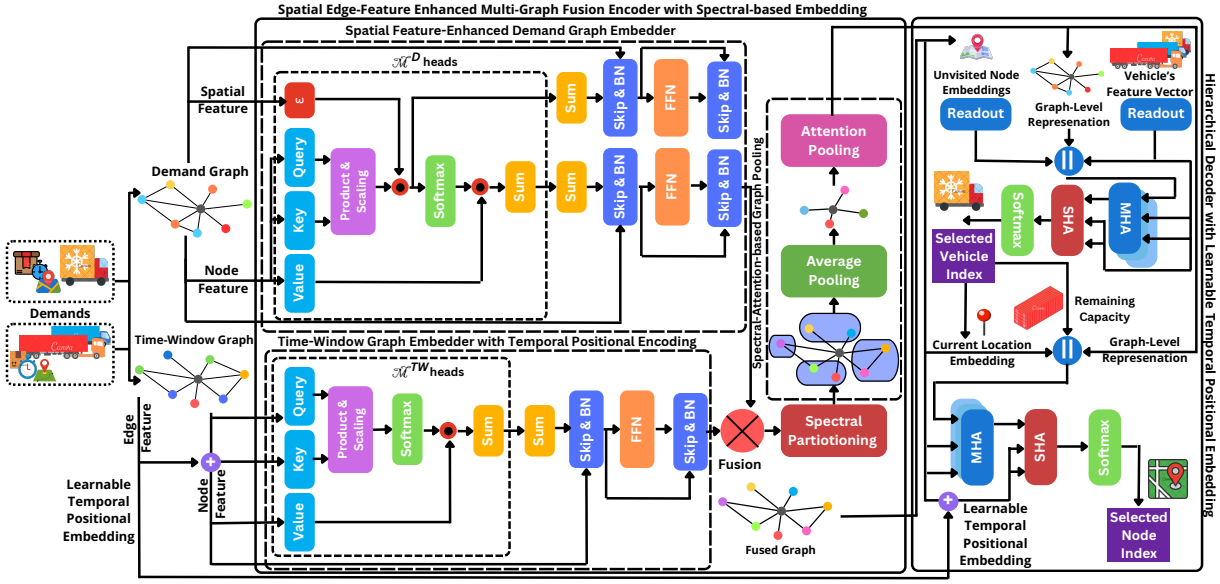


Figure 1: EFECTIW-ROTER's policy network architecture

$$\alpha_{ij,l}^{D,m} = \frac{\exp\left(u_{ij,l}^{D,m}\right)}{\sum_{k\in\mathcal{N}(i)}\exp\left(u_{ik,l}^{D,m}\right)} \times \varepsilon_{e_{ij},l}^{m} \times a_{ij}^{D}, \tag{2}$$

where $\mathcal{N}^D(i)$ denotes the set of nodes connected to node i and d_k is the embedding dimension of the *key*. Incorporating the updated edge feature $\varepsilon_{e_{ij},l}^{m}$ which represents a crucial spatial feature, into the attention weight calculation enables the policy model to effectively based on the road network travel times between nodes. This is particularly vital under time-window constraints where precise and timely customer service is essential. The multiplication of the adjacency matrix entry a_{ij}^{D} and averaging over nodes connected to node i enables the model to efficiently leverage the graph connectivity inductive bias in the demand graph and capture its underlying structure [6], leading to improved routing performance in the presence of demand heterogeneity.

Subsequently, the updated features for node v_i^D at the end l-th layer, i.e. $h_{v_i,l+1}^{D}$, are calculated:

$$h'^{D}{}_{v_i,l} = BN\left(\sum_m \omega_{O,l}^{m}\sum_j \alpha_{ij,l}^{D,m} v_{v_j,l}^{m} + h^{D}{}_{v_i,l}\right), \tag{3}$$

$$h_{v_i,l+1}^{D} = BN\left(FF\left(h'^{D}{}_{v_i,l}\right) + h'^{D}{}_{v_i,l}\right), \tag{4}$$

where $\omega_{O,l}^{m}$ represents a learnable parameter. 'BN' and 'FF' refer to batch normalization and feed-forward layer equipped with a ReLU activation function, respectively. Through learnable parameters $\omega_{O,l}^{m}$, the updated edge features at the end of the l-th layer, $h_{e_{ij},l+1}^{D}$, are calculated:

$$h'^{D}{}_{e_{ij},l} = BN\left(\sum_m \omega_{O,l}^{m} u_{ij,l}^{D,m} + h_{e_{ij},l}^{D}\right), \tag{5}$$

$$h_{e_{ij},l+1}^{D} = BN\left(FF\left(\sum_m \omega'^{m}_{O,l} u_{ij,l}^{D,m}\right) + h'^{D}_{e_{ij},l}\right). \tag{6}$$

4.1.2 *Time-Window Graph Embedder with Temporal Positional Encoding*.

Similar to the demand graph, the embedding process for the time-window graph at the l-th layer initiates with the computation of the *key*, *query*, and *value* for each head m $\left(m \in \{1,\ldots,\mathcal{M}^{TW}\}\right)$. These are respectively calculated using trainable parameters $\omega'^{m}_{Q,l}$, $\omega'^{m}_{K,l}$, and $\omega'^{m}_{V,l}$, resulting in the equations $q'^{m}_{v_i,l} = \omega'^{m}_{Q,l} h_{v_i,l}^{TW}$, $k'^{m}_{v_i,l} = \omega'^{m}_{K,l} h_{v_i,l}^{TW}$, and $v'^{m}_{v_i,l} = \omega'^{m}_{V,l} h_{v_i,l}^{TW}$.

As noted in OR literature [7, 9, 11, 25], temporal ordering based on time-windows provides invaluable information for guiding the search process in effective routing under these constraints. However, the self-attention mechanism is order-invariant and fails to utilize this temporal information [19]. To address this challenge, we propose the use of learnable relative positional encoding [30] for embedding the time-window graph. Given the adjacency matrix entry a_{ij}^{TW} for the l-th layer and m-th head, the attention score $\alpha_{ij,l}^{TW,m}$ using the learnable relative positional embedding represented by vector P_{ij}^{K} is computed as:

$$u_{ij,l}^{TW} = \frac{q'^{m}_{v_i,l}{}^{\top}\left(k'^{m}_{v_i,l} + P_{ij}^{K}\right)}{\sqrt{d_{k'}}}, \tag{7}$$

$$\alpha_{ij,l}^{TW,m} = \frac{\exp\left(u_{ij,l}^{TW,m}\right)}{\sum_{k\in\mathcal{N}'(i)}\exp\left(u_{ik,l}^{TW,m}\right)} \times a_{ij}^{TW}, \tag{8}$$

where $d_{k'}$ denotes the embedding dimension of the *key*. Given the time-window edge feature, as defined in Definition 3.3, the learnable relative position embedding vector P_{ij}^{K} is computed as:

$$P_{ij}^{K} = \epsilon_{ij}^{TW}\omega_{P,l}^{V}, \tag{9}$$

where $\omega^V_{P,l}$ denotes the learnable parameter for the l-th layer, shared among the heads. To enhance memory efficiency in the calculation of learnable relative positional embeddings, we have adopted the techniques proposed by Huang et al. [12].

Finally, the updated features for node v_i at the end l-th layer, i.e. $h^T_{v_i,l+1}W$, are calculated:

$$P^V_{ij} = \epsilon^{TW}_{ij}\omega^V_{P,l}, \tag{10}$$

$$h'^{TW}_{v_i,l} = BN\left(\sum_m \omega^m_{O,l}\sum_j \alpha^{TW,m}_{ij,l}\left(v^m_{v_j,l} + P^V_{ij}\right) + h^{TW}_{v_i,l}\right), \tag{11}$$

$$h^{TW}_{v_i,l+1} = BN\left(FF\left(h'^{D}_{v_i,l}\right) + h'^{TW}_{v_i,l}\right), \tag{12}$$

where $\omega'^{m}_{O,l}$ and $\omega^V_{P,l}$ denote trainable parameters.

Using learnable parameters $\omega'^{m}_{O,l}$, the updated edge features of the time-window graphs at the end of the l-th layer, $h^{TW}_{e_{ij},l+1}$, are computed as follows:

$$h'^{TW}_{e_{ij},l} = BN\left(\sum_m \omega'^{m}_{O,l}u^{TW}_{ij,l} + h^{TW}_{e_{ij},l}\right), \tag{13}$$

$$h^{TW}_{e_{ij},l+1} = BN\left(FF\left(\sum_m \omega'^{m}_{O,l}u^{TW,m}_{ij,l}\right) + h'^{TW}_{e_{ij},l}\right). \tag{14}$$

4.1.3 ***Multi-Graph Fusion***. In this layer, the transformed node features of the demand graph $h^D_{v_i,L}$, for each $v_i \in V^D$, and the time-window graph $h^{TW}_{v_i,L'}$, for each $v_i \in V^{TW}$, are fused using a weighted sum approach. Each feature vector from both graphs is multiplied by a dynamically learned weight that reflects its relevance to the specific task. These weights allow the model to selectively emphasize or de-emphasize features based on their relative importance. This fusion method enables the model to dynamically adjust the contribution of each feature during training, providing a flexible and efficient mechanism for integrating diverse feature sets. By employing a weighted sum, the model maintains the original dimensionality of the features while adaptively moderating the influence of each based on its relevance. Using trainable parameters $\omega^D_{v_i}$ and $\omega^{TW}_{v_i}$, the resulting node features are calculated as follows:

$$h^{fused}_{v_i} = \omega^D_{v_i}h^D_{v_i,L} + \omega^{TW}_{v_i}h^{TW}_{v_i,L'}. \tag{15}$$

4.1.4 ***Spectral-Attention-based Graph Pooling***. Effectively representing the system state is crucial for routing decisions in DRL-based methods. Particularly for the HFDVRPTW, the node connectives is significantly influenced by constraints such as time-windows and heterogeneous customer demands. These factors necessitate a representation that captures both the local and global structures of the problem instance, enabling the model to construct routes effectively in presence of the constraints. To address this need, we propose a spectral-attention-based graph pooling that integrates the strengths of spectral graph clustering and attention mechanisms to generate a comprehensive graph embedding. This module is designed to effectively summarize the complex problem space by leveraging both structural insights and feature relevancy.

The pooling mechanism begins with the construction of a fused graph $G^{fused} = \left(V^{fused}, E^{fused}\right)$, where V^{fused} and E^{fused} denote its node set and edge set, respectively. Each node in the fused graph, denoted by v^{fused}_i, is attributed with features $h^{fused}_{v_i}$, corresponding to the same customer/depot as v^D_i in the demand graph. Given the adjacency matrices of the demand graph $\mathcal{A}^D$ and the time-window graph $\mathcal{A}^{TW}$, the fused graph's adjacency matrix is defined as $\mathcal{A}^{fused} = \mathcal{A}^D \odot \mathcal{A}^{TW}$, where $\odot$ represents the element-wise product. Following this, a spectral graph clustering technique [5], is employed for dimensionality reduction while preserving the topological structure. This step not only preserves the graph's essential topological features but also identifies significant node groupings that are crucial for understanding node connectivity influenced by the constraints. The result of spectral clustering is a partition set $C = \{c_1, \ldots, c_N\}$, where each cluster c_i forms a super-node in the coarsened graph, denoted by $G^{coarse} = (V^{coarse}, E^{coarse})$. Each super-node v^{coarse}_j is assigned a feature vector computed as the average of the fused graph node features within that cluster: $h^{coarse}_{v_j} = \frac{1}{|c_j|}\sum_{v_i \in c_j} h^{fused}_{v_i}$. This aggregation encapsulates the collective characteristics of the nodes within each cluster, facilitating a compact yet informative representation. Finally, we employ a shared-parameter attention-based pooling mechanism to compute the final graph embedding, denoted by h_{graph}. This method employs a learnable attention mechanism to dynamically assign weights to the importance of each node within the coarsened graph, thereby enabling the model to selectively emphasize the most significant clusters for generating a graph-level representation.

$$z_{v_i} = \omega^1 \tanh\left(\omega^2 h^{coarse}_{v_i} + b\right), \forall v_i \in V^{coarse}, \tag{16}$$

$$h_{graph} = \sum_{v_i \in V^{coarse}} h^{coarse}_{v_i} \cdot softmax\left(z_{v_i}\right), \tag{17}$$

where ω^1, ω^2, and b are trainable parameters.

4.2 Hierarchical Decoder with Learnable Temporal Positional Embedding

Given the vehicle feature vector $\mathcal{F}$, the fused graph's node embeddings $h^{fused}_{v_i}, \forall v_i \in V^{fused}$, and the graph-level representation h_{graph}, the decoder operates in two distinct stages at each decoding step $\mathcal{T}$. Initially, it computes a probability vector $p^{\mathcal{F}}_{\mathcal{T}}$, which represents the likelihood of each vehicle in the heterogeneous fleet being selected to extend its route at this step. Subsequently, the decoder generates a probability distribution $p^V_{\mathcal{T}}$, which indicates the probability that each node v^{fused}_i will be the next customer node to be visited by the selected vehicle.

For the vehicle selection stage, given the vehicle feature vector $\mathcal{F}$ (defined in Definition 3.1) and the vehicles state $s^{\mathcal{F}}_{\mathcal{T}}$ (defined in Section 3) for each vehicle k, a vehicle representation is computed as: $\hat{h}^{f_k}_{\mathcal{T}} = FF\left[FF\left(f_k\right) \parallel rc^k_{\mathcal{T}} \parallel h^{fused}_{V^k_{\mathcal{T}}}\right]$. This results in the vehicles embedding vector $\hat{H}^{\mathcal{F}}_{\mathcal{T}} = \left\{\hat{h}^{f_1}_{\mathcal{T}}, \ldots, \hat{h}^{f_\gamma}_{\mathcal{T}}\right\}$. FF denotes a two-layer fully connected neural network with ReLU activation functions. To enhance computational efficiency, FF is applied to f_k only once at

the beginning of each episode. Consequently, given the vehicles' embedding vector $\hat{H}^{\mathcal{F}}_{\mathcal{T}}$, the graph-level representation h_{graph}, the routing state $s^{\mathcal{R}}_{\mathcal{T}}$, and the fused graph's node embeddings $h^{fused}_{v_i}$ $\forall v_i \in V^{fused}$, a vehicle context embedding $h^{(c)^{\mathcal{F}}}_{\mathcal{T}}$ is calculated as follows:

$$h^{(c)^{\mathcal{F}}}_{\mathcal{T}} = \left[h_{graph} \parallel \text{readout}\left(\hat{H}^{\mathcal{F}}_{\mathcal{T}}\right) \parallel \text{readout}\left(\bigcup_{v_i \in Vfused \setminus s^{\mathcal{R}}_{\mathcal{T}}} h^{fused}_{v_i} \right) \right], \quad (18)$$

where 'readout' refers to the standard readout operation [2]. $\parallel$ denotes the concatenation operation. This context embedding is designed to outline the problem at hand, the current status of the vehicle fleet, and the remaining nodes, providing a comprehensive basis for informed vehicle selection. This ensures that the policy model makes decisions based on an understanding of the available resources and customers' demands.

Subsequently, using the vehicle embeddings $\hat{H}^{F}$ along with the context embedding $h^{(c)^{\mathcal{F}}}_{\mathcal{T}}$, an embedding $h^{(g)^{\mathcal{F}}}_{\mathcal{T}}$ is generated through a Multi-Head Attention (MHA) mechanism, following the methodology outlined by Vaswani et al. [31]. This process is outlined as $h^{(g)^{\mathcal{F}}}_{\mathcal{T}} = \text{MHA}\left(\omega^{(g)^{\mathcal{F}}}_{Q} h^{(c)^{\mathcal{F}}}_{\mathcal{T}}, \omega^{(g)^{\mathcal{F}}}_{K} \hat{H}^{\mathcal{F}}, \omega^{(g)^{\mathcal{F}}}_{V} \hat{H}^{\mathcal{F}}\right)$, where $\omega^{(g)^{\mathcal{F}}}_{Q}$, $\omega^{(g)^{\mathcal{F}}}_{K}$, and $\omega^{(g)^{\mathcal{F}}}_{V}$ represent the trainable weights for the *query*, *key*, and *value*, respectively. Following that, employing a Single-Head Attention [31], the embedding $h^{(g)^{\mathcal{F}}}_{\mathcal{T}}$ and the vector $\bar{H}^{\mathcal{F}}_{\mathcal{T}}$ alongside with the learnable parameters $\omega^{(c)^{\mathcal{F}}}_{Q}$, $\omega^{(c)^{\mathcal{F}}}_{K}$, and $\omega^{(c)^{\mathcal{F}}}_{V}$ for *query*, *key*, and *value*, respectively, the compatibility $\bar{h}^{(c)^{\mathcal{F}}}_{\mathcal{T}}$ is computed as: $\bar{h}^{(c)^{\mathcal{F}}}_{\mathcal{T}} = \text{SHA}\left(W^{(c)^{\mathcal{F}}}_{Q} h^{(g)^{\mathcal{F}}}_{\mathcal{T}}, W^{(c)^{\mathcal{F}}}_{K} \bar{H}^{\mathcal{F}}_{\mathcal{T}}, W^{(c)^{\mathcal{F}}}_{V} \bar{H}^{\mathcal{F}}_{\mathcal{T}}\right)$. Ultimately, a probability distribution $p^{f}_{\mathcal{T}}$ over the heterogeneous fleet is derived via a Softmax function applied to $\bar{h}^{(c)^{\mathcal{F}}}_{\mathcal{T}}$.

Subsequent to the vehicle selection from the heterogeneous fleet, with vehicle index k identified in the vehicle selection stage, the route selection stage begins by forming a context embedding $h^{(c)^{\mathcal{R}}}_{\mathcal{T}}$. Considering the state of the selected vehicle $s^{f_k}_{\mathcal{T}}$, the graph-level representation h_{graph}, and the fused graph's node embeddings vector $H^{fused} = \left[h^{fused}_{v_0}, \ldots, h^{fused}_{v_C}\right]$, the context embedding is constructed as: $h^{(c)^{\mathcal{R}}}_{\mathcal{T}} = \left[rc^{k}_{\mathcal{T}} \parallel h^{fused}_{V^{k}_{\mathcal{T}}} \parallel h_{graph} \right]$. This context embedding provided the policy model with a comprehensive yet concise overview of the underlying structure of the HFDVRPTW instance targeted to solved. It also incorporates the current state of the vehicle designated to extend its route at the current decoding step.

The context embedding $h^{(c)^{\mathcal{R}}}_{\mathcal{T}}$, together with the fused graph's node embeddings vector H^{fused}, is then utilized by the Multi-Head Attention (MHA) mechanism to compute a *glimpse* of the most relevant nodes, enabling the model to focus on the most informative parts of the graph. With trainable parameters $\omega'^{(g)}_{Q}$, $\omega'^{(g)}_{K}$, and $\omega'^{(g)}_{V}$, the glimpse $h^{(g)^{\mathcal{R}}}_{\mathcal{T}}$ is computed as follows:

$$h^{(g)^{\mathcal{R}}}_{\mathcal{T}} = MHA\left(\omega^{(g)^{\mathcal{R}}}_{Q} h^{(c)^{\mathcal{R}}}_{\mathcal{T}}, \omega^{(g)^{\mathcal{R}}}_{K} \hat{H}^{\mathcal{R}}, \omega^{(g)^{\mathcal{R}}}_{V} \hat{H}^{\mathcal{R}}\right). \quad (19)$$

Next, a compatibility score for each node v^{fused}_{i} in the fused graph is computed using a compatibility layer. This involves calculating the alignment between the node embedding $h^{fused}_{v_i}$ and the glimpse $h^{(g)^{\mathcal{R}}}_{\mathcal{T}}$ using trainable parameters $\omega^{(u)}_{Q}$ and $\omega^{(u)}_{K}$. The process is outlined as follows:

$$u^{\mathcal{T}}_{(c),i} = \begin{cases} C.tanh\left(\dfrac{\left(\omega^{(u)}_{Q} h^{(g)}_{\mathcal{T}}\right)^{\intercal} \left(\omega^{(u)}_{K}\left(h^{fused}_{v_i} + P'^{K}_{V^{k}_{\mathcal{T}} i}\right)\right)}{\sqrt{d_k}} \right), & \text{if } M_{i,\mathcal{T}} = 0, \\ -\infty, & \text{otherwise,} \end{cases} \quad (20)$$

where C is a constant used to control entropy. The masking vector $M_{i,T}$, associated with node v^{fused}_{i}, is defined as follows:

$$M_{i,\mathcal{T}} = \begin{cases} 1, & \text{if } v^{fused}_{i} = v^{fused}_{0} \text{ and } h^{fused}_{V^{k}_{\mathcal{T}}} = v^{fused}_{0}, \\ 1, & \text{if } v^{fused}_{i} \in s^{\mathcal{R}}_{\mathcal{T}}, \\ 1, & \text{if } rc^{k}_{\mathcal{T}} < d_i \text{ (detailed in Definition 3.2)}, \\ 1, & \text{if } a^{D}_{V^{k}_{\mathcal{T}} j} = 0 \text{ or } a^{TW}_{V^{k}_{\mathcal{T}} i} = 0, \\ 0, & \text{otherwise,} \end{cases} \quad (21)$$

The first rule ensures that the depot is not visited in two consecutive decoding steps. The second rule prevents a customer from being visited more than once. The third rule ensures compliance with the vehicle's maximum capacity. The fourth rule ensures the reachability of the current node visited by vehicle k to node v^{fused}_{i}, based on the corresponding entries in the demand graph and time-window graph adjacency matrices. The positional embedding $P'^{K}_{V^{k}_{\mathcal{T}} i}$ is computed using the trainable parameter $\omega'^{K}_{P,l}$ and the time-window graph edge feature $\epsilon^{TW}_{V^{k}_{\mathcal{T}} i}$ (detailed in Definition 3.3), as follows:

$$P'^{K}_{V^{k}_{\mathcal{T}} j} = \epsilon^{TW}_{V^{k}_{\mathcal{T}} i} \omega'^{K}_{P,l}. \quad (22)$$

The addition of the positional embedding $P^{K}_{V^{k}_{\mathcal{T}} j}$ enables the decoder to make informed decisions based on the ordering of time-windows and the current location of the vehicle, i.e $V^{k}_{\mathcal{T}}$, thereby facilitating effective routing under time-window constraints.

Finally, the probability vector $p^{V}_{\mathcal{T}}$ is computed using the softmax function applied to the compatibility scores $u^{\mathcal{T}}_{(c),i}$ through:

$$p^{V}_{\mathcal{T}} = \text{softmax}\left(u^{\mathcal{T}}_{(c),i}\right). \quad (23)$$

4.3 Training Algorithm

To train the Transformer-style policy network of EFECTIW-ROTER, the REINFORCE reinforcement learning algorithm [33] has been employed. This method adopts a policy-gradient strategy complemented by a greedy roll-out baseline [27]. Feedback for adjusting the model's parameters during training is provided by the cumulative travel time of the routes generated at the end of each episode. It is noteworthy that our method does not generate infeasible solution but it might not be able to find the feasible solution for instances especially those with strict constraints.

5 EXPERIMENTS

In this section, we present the experimental analyses on two real-world datasets based on traffic data from two major Canadian cities, Calgary and Edmonton, to address the following research questions:

- RQ1: How does EFECTIW-ROTER perform solving the HFD-VRPTW in comparison with state-of-the-art DRL-based and heuristic methods, particularly in terms of total travel time and computational efficiency?
- RQ2: How does EFECTIW-ROTER generalize when solving HFDVRPTW instances of different sizes?
- RQ3: What are the contributions of spatial feature enhancement, time-window graph embedding, temporal positional encoding in the encoder, spectral-attention-based graph pooling, and temporal positional embedding in the decoder to minimize total travel time?

5.1 Experimental Settings

5.1.1 Equipment and hyperparameters. We conducted training and testing experiments for DRL-based models on servers featuring V100 GPUs. For all other baseline models, we conducted experiments on servers equipped with 32 cores of Intel(R) Xeon(R) Gold 6240R CPUs. The settings for our DRL-based models include a hidden dimension size of 128, 3 encoding layers (including embedding graphs for demand and time-windows), and 8 attention heads. These models are trained on 512,000 problem instances using the Adam optimizer for 500 epochs, with a batch size of 256 and a learning rate of 10^{-3}. All the methods are tested on 2,000 instances.

5.1.2 Datasets. The experiments are carried out on three different problem sizes: 20, 50, and 100. Following Kool et al. [15], in these sets, customer demands are randomly selected from integers ranging from 1 to 9. Locations are uniformly selected from Calgary and Edmonton in Canada. The estimated travel time between each source and destination (i.e., the depot and the customers) is calculated based on the total distance traveled on each road segment and the estimated travel speed of that segment (calculated using GPS records from the working day with the highest number of records over a 4-hour period). For generating demands' time-window, we employ the method used by Lin et al. [20]. The fleet comprises vehicles with capacities of 20, 30, and 40 for problems of size 20. For problems of size 50, the available vehicle capacities are 30, 40, and 50, while for size 100, the options are 40, 50, and 60. Each of these vehicles can be either refrigerated or non-refrigerated. For determining which vehicles can serve the customers, two numbers are uniformly sampled. The first number, ranging from 1 to 3, denotes the starting point, and the second, ranging from the first number to 3, indicates the endpoint among the three capacities. For the type, each has an equal chance of being chosen uniformly. For problem sizes of 100, there are 2 vehicles for each attribute and 1 vehicle for the remaining problem sizes, specifically 20 and 50.

5.1.3 Baselines. To evaluate EFECTIW-ROTER, we utilized the following state-of-the-art DRL-based and heuristic methods:

- **ACO:** Improved ant colony optimization (ACO) algorithm proposed by Han et al. [10] for solving the heterogeneous fleet vehicle routing problem with time-windows (HFDVRPTW).
- **GA:** A genetic algorithm (GA) integrating the method used by Zhu et al. [38] for handling time-windows with the approach used by Kang et al. [13] for managing heterogeneity.
- **VNS/TS:** Variable neighborhood search / tabu search (VNS/TS) [28] combines VNS's neighborhood changes with Tabu Search's memory strategy, exploring multiple neighborhoods and using a tabu list to avoid revisiting solutions.
- **Sweep:** The sweep algorithm [26] groups customers into routes by systematically sweeping around a central depot based on their angular coordinates.
- **VRP-RL [24]:** A DRL-based approach featuring an RNN-based policy network.
- **AM [15]:** A state-of-the-art DRL method utilizing a Transformer-style architecture for its policy model.
- **EVRPTW-DRL [20]:** A DRL approach that combines a Transformer-style architecture with an RNN, tailored for time-window constraints.
- **DRL-TS [3]:** A DRL method featuring a policy model with an RNN and Transformer-based decoder, and two encoders, aiming to minimize total road network travel time.

Since DRL-based methods lack the ability to handle demand heterogeneity, the decoder is allowed to select one of the eligible vehicle at at each decoding step, while the rest are masked. In the experiments conducted on EFECTIW-ROTER, AM, EVRPTW-DRL, and DRL-TS, two distinct testing strategies were utilized. The first strategy, called Greedy, involves choosing the vehicle and node with the highest probability at each decoding step by the vehicle selection and trip construction decoders. The second strategy, known as Sampling, involves generating 1280 solutions based on the probability distributions produced by the decoding modules and then selecting the best solution from these samples.

5.1.4 Evaluation Metrics. In this study, performance is evaluated using total travel time, running time, and the performance gap. The gap, assuming $Cost^{best}$ is the minimal cost achieved, is defined as:

$$Gap = \frac{Cost - Cost^{best}}{Cost^{best}} \times 100\%. \tag{24}$$

5.2 Performance Comparison (RQ1)

In this section, we aim to compare the performance of EFECTIW-ROTER with baseline methods, including competitive conventional heuristics and state-of-the-art DRL-based approaches, in terms of cost and computation time. The results for instances from Calgary dataset are presented in Table 1. We summarized the experimental results for Edmonton dataset in Appendix B.

For the Calgary dataset, in terms of cost, as shown in Table 1, EFECTIW-ROTER with the *Greedy*-based decoding strategy, i.e., EFECTIW-ROTER (Greedy), significantly outperforms Sweep heuristics across all problem instance sizes, with a notable gap exceeding 33.04%. Additionally, EFECTIW-ROTER (Greedy) surpasses all DRL-based approaches using the same decoding strategy, as well as VRP-DRL, in average total travel time of generated routes for all problem sizes, while requiring less computational time compared to methods with RNN-based policy networks, such as VRP-DRL, EVRPTW-DRL, and DRL-TS. EFECTIW-ROTER (Greedy) also surpasses ACO heuristics and all DRL-based methods with

Table 1: EFECTIW-ROTER vs. baselines for solving HFDVRPTW (Calgary dataset). The ↓ indicates the lower value is better.

	Calgary Dataset								
	HFEVRPTW-20			HFEVRPTW-50			HFEVRPTW-100		
Model	Cost ↓	Gap ↓	Time ↓	Cost ↓	Gap ↓	Time ↓	Cost ↓	Gap ↓	Time ↓
Sweep	742.2	33.04%	2.9 mins	1520.2	41.62%	12.3 mins	3734.3	53.35%	32.9 mins
ACO	577.5	3.53%	12.7 mins	1207.4	12.48%	55.2 mins	2876.2	18.94%	3.4 hs
GA	577.2	3.46%	10.9 mins	1183.4	10.24%	2.1 hs	2785.3	14.38%	17.1 hs
VNS/TS	562.9	0.91%	12.3 mins	1127.1	5.00%	1.3 hs	2680.4	10.07%	13.4 hs
VRP-DRL	623.9	11.84%	2.13 s	1279.9	19.23%	4.38 s	2980.4	22.39%	9.26 s
AM (Greedy)	616.8	10.56%	1.25 s	1253.7	16.79%	2.47 s	2965.0	21.76%	5.46 s
AM (Sampling)	580.4	4.05%	9.8 mins	1200.0	11.79%	39.7 mins	2827.7	16.12%	1.7 hs
EVRPTW-DRL (Greedy)	616.3	10.47%	2.74 s	1232.8	14.84%	5.24 s	2906.1	19.34%	11.19 s
EVRPTW-DRL (Sampling)	578.9	3.78%	17.8 mins	1194.0	11.23%	1.5 hs	2821.4	15.86%	4.2 hs
DRL-TS (Greedy)	615.6	10.35%	2.07 s	1226.6	14.27%	4.43 s	2897.4	18.98%	8.74 s
DRL-TS (Sampling)	579.2	3.83%	13.92 mins	1192.8	11.12%	1.1 hs	2808.7	15.34%	2.8 hs
EFECTIW-ROTER (Greedy) (ours)	585.1	4.89%	1.78 s	1190.1	10.87%	3.15 s	2635.6	8.23%	7.83 s
EFECTIW-ROTER (Sampling) (ours)	**557.9**	**0.00%**	12.4 mins	**1073.5**	**0.00%**	42.1 mins	**2435.2**	**0.00%**	2.3 hs

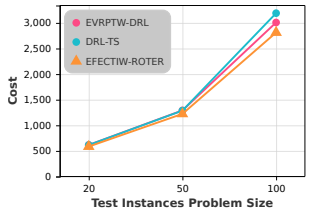
Sampling-based decoding for problem sizes larger than 50, while significantly reducing computation time. Additionally, EFECTIW-ROTER (Greedy) outperforms competitive GA and VNS/TS methods in total travel time for problem sizes of 100.

EFECTIW-ROTER (Sampling) demonstrates much better performance in cost compared to all the baselines, the gap in cost increases as the problem size grows. In terms of computational costs, EFECTIW-ROTER (Sampling) is faster than DRL-based methods with RNN-based policy models and with a similar decoding strategy for all problem instances. Compared to the competitive heuristics, EFECTIW-ROTER (Sampling) is faster in computation for problems sizes over 50 and works competitively on problem instances of size 20 in running time.

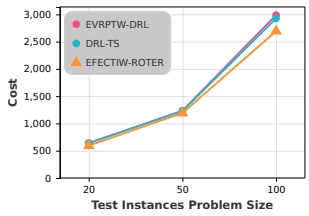
Performance evaluation on the Edmonton dataset shows similar results. Overall, the experiments on both datasets demonstrate the effectiveness of EFECTIW-ROTER in solving the routing problem targeted in this paper, i.e., HFEVRPTW.

5.3 Generalizability (RQ2)

Given the intractability of training the model on large datasets in industrial environments due to computational and resource limitations, the model's ability to generalize to larger-scale problems becomes crucial. Therefore, in this section, we investigate the generalizability of EVRPTW-DRL, DRL-TS, and EFECTIW-ROTER by training the policy models on instances of sizes 20 and 50 from the Calgary dataset and testing them on instances with 20, 50, and 100 customers with *Greedy* decoding strategy. The results of this experimental study are summarized in Figure 2. According to the experimental results, for all of these DRL-based methods, the best performance is achieved when the policy model is trained and tested on the same problem sizes. However, the performance loss can be acceptable when prioritizing shorter training times. The results also demonstrate that EFECTIW-ROTER performs better compared to EVRPTW-DRL and DRL-TS, with the performance gap widening (indicated by increasing differences between the y values) as the difference between the problem instance sizes for training and testing increases. In fact, for the models trained on instances of size 20, the performance gap favoring EFECTIW-ROTER over other methods is more than 5.2% when tested on size 50 instances and over 6.9% when tested on size 100 instances. When the models are trained on instances with 50 customers, the performance Gap favoring EFECTIW-ROTER over the two other DRL-based methods is more that 8.6%. These results indicate the generalization ability of EFECTIW-ROTER in solving HFEVRPTW instances that are larger than those on which the policy model was trained, capability critical for applicability in real-world operations.



(a) Trained on instances of size 20



(b) Trained on instances of size 50

Figure 2: EFECTIW-ROTER's generalizability study on Calgary dataset

5.4 Ablation Study (RQ3)

To improve DRL-based method for effectively solving HFEVRPTW, five major enhancements have been made to the policy model:

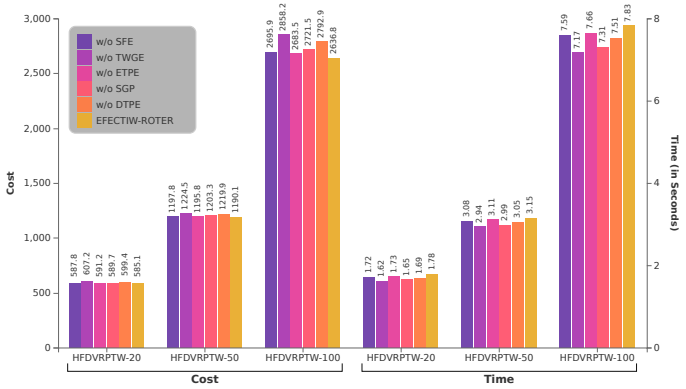


Figure 3: EFECTIW-ROTER's ablation study on Calgary dataset

spatial feature enhancement (SPE), time-window graph embedding (TWGE), temporal positional encoding in the encoder (ETPE), spectral-attention-based graph pooling (SGP), and temporal positional embedding in the decoder (DTPE). In this section, we aim to investigate the impact of each of these enhancements an ablation study of EFECTIW-ROTER with the Greedy decoding strategy on the Calgary dataset with three different problem instance sizes, where the results are illustrated in Figure 3.

As evident from Figure 3, all five enhancements play a role in improving the cost, underscoring the rationale for the design of the policy model. The largest performance gap in cost, resulting from the removal of TWGE (over 7.7 percent for HFDVRPTW-100), highlights the importance of this module in capturing the underlying structure of the problem based on time-window constraints, despite its significant share in running time (up to 0.66 s for instances with 100 customers). Following TWGE, DTPE shows the second-highest contribution to performance improvement in terms of cost, with a 5.5% enhancement. This significant improvement underscores the importance of capturing temporal ordering based on time-windows and utilizing it for enhanced routing in the presence of this constraint. Among these five improvements, ETPE exhibits the lowest contribution (just over 1.7 percent for HFDVRPTW-100); nevertheless, this enhancement also places the least demand on computational resources. The contribution of SFE and SGP to minimizing the cost is similar, each exceeding 2 percent for instances with 100 customers. This validates the effectiveness of SGP in representing the problem instance for efficient routing. This further highlights the capability of EFECTIW-ROTER in exploiting road network travel time for routing enhancement.

6 CONCLUSION

In this study, we investigate solving the HFEVRPTW, an optimization problem of great importance for real-world logistics operations. We introduce a DRL-based method named EFECTIW-ROTER, which features a Transformer-style policy network. EFECTIW-ROTER's encoder leverages a multi-head attention fusion encoder network with a graph connectivity inductive bias, based on the connectivity structure imposed by time-window constraints and demand heterogeneity. The encoder captures the road network travel times between the nodes for enhanced routing. Through a spectral-attention-based graph pooling mechanism, EFECTIW-ROTER's encoder effectively reflects the underlying structure of the problem in a graph-level embedding, enabling effective decision-making by the policy model based on the problem instance. EFECTIW-ROTER employs a hierarchical attention decoder to account for the fleet's heterogeneity. Equipped with positional embeddings, the decoder is empowered to make effective routing decisions based on the time-window constraints. Experimental results on real-world traffic data from two major Canadian cities, Calgary and Edmonton, demonstrate that EFECTIW-ROTER not only outperforms current state-of-the-art DRL-based and heuristic methods in reducing travel times but also achieves faster computational times compared to heuristics, while also generalizing well to larger-scale problems.

Regarding avenues for future research, there are a few directions we aim to pursue. Firstly, we aim to broaden the scope of our research by incorporating variations that involve mixed pickup and delivery scenarios, which are particularly relevant in real-world contexts like beverage distribution. Secondly, we plan to investigate routing strategies that take into account the possibility of order cancellations, thereby accommodating the stochastic nature that may exist in certain logistics applications. Additionally, with the increasing adoption of medium- and heavy-duty electric vehicles as part of carbon-neutral policies in modern countries, we intend to focus on route optimization specifically tailored for electric vehicles. This will involve addressing the unique challenges associated with integrating these environmentally friendly vehicles into existing fleet operations.

ACKNOWLEDGMENTS

The authors would like to express their gratitude to Dr. Yani Ioannou for his invaluable suggestions and many fruitful discussions. We also extend our thanks to Maryam Rezaie for her contributions in enhancing the visualizations presented. This project was supported in part by collaborative research funding from the National Research Council of Canada's Artificial Intelligence for Logistics Program. The authors also would like to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) for a part of financial support.

REFERENCES

[1] Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. 2017. Neural Combinatorial Optimization with Reinforcement Learning. https://openreview.net/forum?id=rJY3vK9eg

[2] David Buterez, Jon Paul Janet, Steven J Kiddle, Dino Oglic, and Pietro Liò. 2022. Graph neural networks with adaptive readouts. *Advances in Neural Information Processing Systems* 35 (2022), 19746–19758.

[3] Jinbiao Chen, Huanhuan Huang, Zizhen Zhang, and Jiahai Wang. 2022. Deep reinforcement learning with two-stage training strategy for practical electric vehicle routing problem with time windows. In *International conference on parallel problem solving from nature*. Springer, 356–370.

[4] Lu Duan, Yang Zhan, Haoyuan Hu, Yu Gong, Jiangwen Wei, Xiaodong Zhang, and Yinghui Xu. 2020. Efficiently solving the practical vehicle routing problem: A novel joint learning approach. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 3054–3063.

[5] Alexandre Duval and Fragkiskos Malliaros. 2022. Higher-order clustering and pooling for graph neural networks. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 426–435.

[6] Vijay Prakash Dwivedi and Xavier Bresson. 2021. A Generalization of Transformer Networks to Graphs. *AAAI Workshop on Deep Learning on Graphs: Methods and Applications* (2021).

[7] Richard W Eglese, Alan Mercer, and Babak Sohrabi. 2005. The grocery superstore vehicle scheduling problem. *Journal of the Operational Research Society* 56 (2005), 902–911.

[8] Vikas Garg, Stefanie Jegelka, and Tommi Jaakkola. 2020. Generalization and representational limits of graph neural networks. In *International Conference on Machine Learning*. PMLR, 3419–3430.

[9] Oscar M González, Carlos Segura, and Sergio I Valdez Peña. 2018. A parallel memetic algorithm to solve the capacitated vehicle routing problem with time windows. *International Journal of Combinatorial Optimization Problems and Informatics* 9, 1 (2018), 35.

[10] Jihee Han, Arash Mozhdehi, Yunli Wang, Sun Sun, and Xin Wang. 2022. Solving a multi-trip vrp with real heterogeneous fleet and time windows based on ant colony optimization: An industrial case study. In *Proceedings of the 15th ACM SIGSPATIAL International Workshop on Computational Transportation Science*. 1–4.

[11] Chaug-Ing Hsu, Sheng-Feng Hung, and Hui-Chieh Li. 2007. Vehicle routing problem with time-windows for perishable food delivery. *Journal of food engineering* 80, 2 (2007), 465–475.

[12] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam M. Shazeer, Ian Simon, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. 2018. Music Transformer: Generating Music with Long-Term Structure. In *International Conference on Learning Representations*. https://api.semanticscholar.org/CorpusID:54477714

[13] He-Yau Kang and Amy HI Lee. 2018. An enhanced approach for the multiple vehicle routing problem with heterogeneous vehicles and a soft time window. *Symmetry* 10, 11 (2018), 650.

[14] AJ Klein. 1987. Microcomputer-based vehicle routing and scheduling: An overview. (1987).

[15] Wouter Kool, Herke van Hoof, and Max Welling. 2018. Attention, Learn to Solve Routing Problems!. In *International Conference on Learning Representations*.

[16] Keyju Lee and Junjae Chae. 2021. A proposal and analysis of new realistic sets of benchmark instances for vehicle routing problems with asymmetric costs. *Applied Sciences* 11, 11 (2021), 4790.

[17] Keyju Lee and Junjae Chae. 2023. Estimation of Travel Cost between Geographic Coordinates Using Artificial Neural Network: Potential Application in Vehicle Routing Problems. *ISPRS International Journal of Geo-Information* 12, 2 (2023), 57.

[18] Jingwen Li, Yining Ma, Ruize Gao, Zhiguang Cao, Andrew Lim, Wen Song, and Jie Zhang. 2021. Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem. *IEEE Transactions on Cybernetics* 52, 12 (2021), 13572–13585.

[19] Yang Li, Si Si, Gang Li, Cho-Jui Hsieh, and Samy Bengio. 2021. Learnable fourier features for multi-dimensional spatial positional encoding. *Advances in Neural Information Processing Systems* 34 (2021), 15816–15829.

[20] Bo Lin, Bissan Ghaddar, and Jatin Nathwani. 2021. Deep reinforcement learning for the electric vehicle routing problem with time windows. *IEEE Transactions on Intelligent Transportation Systems* 23, 8 (2021), 11528–11538.

[21] Liheng Ma, Chen Lin, Derek Lim, Adriana Romero-Soriano, Puneet K Dokania, Mark Coates, Philip Torr, and Ser-Nam Lim. 2023. Graph inductive biases in transformers without message passing. In *International Conference on Machine Learning*. PMLR, 23321–23337.

[22] Odalric-Ambrym Maillard, Daniil Ryabko, and Rémi Munos. 2011. Selecting the state-representation in reinforcement learning. *Advances in Neural Information Processing Systems* 24 (2011).

[23] Arash Mozhdehi, Mahdi Mohammadizadeh, and Xin Wang. 2024. Edge-DIRECT: A Deep Reinforcement Learning-based Method for Solving Heterogeneous Electric Vehicle Routing Problem with Time Window Constraints. *arXiv preprint arXiv:2407.01615* (2024).

[24] Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takác. 2018. Reinforcement learning for solving the vehicle routing problem. *Advances in neural information processing systems* 31 (2018).

[25] Michael Polacek, Richard F Hartl, Karl Doerner, and Marc Reimann. 2004. A variable neighborhood search for the multi depot vehicle routing problem with time windows. *Journal of heuristics* 10 (2004), 613–627.

[26] Christian Prins. 2002. Efficient heuristics for the heterogeneous fleet multitrip VRP with application to a large-scale real case. *Journal of Mathematical Modelling and Algorithms* 1, 2 (2002), 135–150.

[27] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7008–7024.

[28] Michael Schneider, Andreas Stenger, and Dominik Goeke. 2014. The electric vehicle-routing problem with time windows and recharging stations. *Transportation science* 48, 4 (2014), 500–520.

[29] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-Attention with Relative Position Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Association for Computational Linguistics.

[30] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-Attention with Relative Position Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, Marilyn Walker, Heng Ji, and Amanda Stent (Eds.). Association for Computational Linguistics, New Orleans, Louisiana, 464–468. https://doi.org/10.18653/v1/N18-2074

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[32] Conghui Wang, Zhiguang Cao, Yaoxin Wu, Long Teng, and Guohua Wu. 2024. Deep Reinforcement Learning for Solving Vehicle Routing Problems With Backhauls. *IEEE Transactions on Neural Networks and Learning Systems* (2024).

[33] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8 (1992), 229–256.

[34] David H Wolpert and William G Macready. 1997. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation* 1, 1 (1997), 67–82.

[35] Yunqiu Xu, Meng Fang, Ling Chen, Gangyan Xu, Yali Du, and Chengqi Zhang. 2021. Reinforcement learning with multiple relational attention for solving vehicle routing problems. *IEEE Transactions on Cybernetics* 52, 10 (2021), 11107–11120.

[36] Hongyu Zang, Xin Li, and Mingzhong Wang. 2022. Simsr: Simple distance-based state representations for deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 8997–9005.

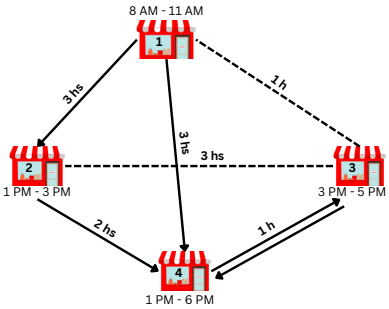
[37] Zhen Zhang, Jiajun Bu, Martin Ester, Jianfeng Zhang, Chengwei Yao, Zhi Yu, and Can Wang. 2019. Hierarchical graph pooling with structure learning. *arXiv preprint arXiv:1911.05954* (2019).

[38] Kenny Qili Zhu. 2000. A new genetic algorithm for VRPTW. In *Proceedings of the international conference on artificial intelligence*, Vol. 311264.

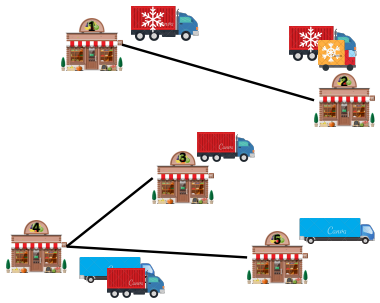
A TIME-WINDOW AND DEMAND GRAPHS GENERATION

In this section, we describe the process of generating time-window and demand graphs. The generation of time-window graphs is illustrated through an example in Figure 4a. As shown in the figure, customer node 3 is reachable from node 4 because there is an overlap between the 1 PM to 6 PM window of node 4 and the 3 PM to 5 PM window of node 3, allowing for a 1-hour travel time. In contrast, customers 1 and 3 are not reachable from each other, as their corresponding time-windows do not overlap when considering the 1-hour travel time between them.

Figure 4b depicts the demand graph generation process. As shown in the figure, customer nodes are connected only if they share the same preference for both vehicle type and size. For instance, nodes 1 and 2 are connected because they have a common



(a) Time-window graph generation: Solid lines show connectivity between the nodes, while dashed lines denote the absence of connectivity.



(b) Demand graph generation: Solid lines show connectivity between the nodes.

Figure 4: Time-window and demand graph generation processes.

truck type and size in their preference lists. In contrast, nodes 2 and 3 are not connected, as, despite requesting the same vehicle size, they differ in their preferred vehicle type.

B PERFORMANCE COMPARISON ON EDMONTON DATASET

	Edmonton Dataset								
	HFEVRPTW-20			HFEVRPTW-50			HFEVRPTW-100		
Model	Cost ↓	Gap ↓	Time ↓	Cost ↓	Gap ↓	Time ↓	Cost ↓	Gap ↓	Time ↓
Sweep	595.1	32.91%	3.1 mins	1371.8	42.18%	12.7 mins	2866.5	49.87%	35.2 mins
ACO	458.5	2.39%	12.1 mins	1080.0	11.93%	59.6 mins	2254.3	17.86%	3.3 hs
GA	459.0	2.52%	11.2 mins	1059.7	9.83%	2.2 hs	2156.1	12.73%	17.5 hs
VNS/TS	448.3	0.12%	12.6 mins	1029.4	6.69%	1.5 hs	2125.3	11.12%	13.2 hs
VRP-DRL	496.3	10.85%	2.09 s	1145.4	18.71%	2.53 s	2368.1	23.81%	9.37 s
AM (Greedy)	491.9	9.87%	1.33 s	1130.0	17.12%	2.39 s	2352.2	22.98%	5.12 s
AM (Sampling)	469.8	4.93%	9.9 mins	1070.4	10.94%	41.9 mins	2236.5	16.93%	1.6 hs
EVRPTW-DRL (Greedy)	488.2	9.04%	2.98 s	1120.4	16.12%	5.05 s	2297.3	20.11%	11.37 s
EVRPTW-DRL (Sampling)	467.7	4.46%	17.1 mins	1070.8	10.98%	1.7 hs	2201.3	15.09%	4.0 hs
DRL-TS (Greedy)	489.0	9.21%	2.28 s	1118.6	15.93%	4.51 s	2265.0	18.42%	8.68 s
DRL-TS (Sampling)	469.0	4.75%	13.65 mins	1067.0	10.59%	1.1 hs	2205.1	15.29%	2.9 hs
EFECTIW-ROTER (Greedy) (ours)	472.2	5.45%	1.84 s	1059.9	9.85%	3.36 s	2080.8	7.79%	7.98 s
EFECTIW-ROTER (Sampling) (ours)	**447.8**	**0.00%**	12.9 mins	**964.9**	**0.00%**	41.9 mins	**1912.7**	**0.00%**	2.6 hs