

Article

Research on Distribution Optimization Strategy of Front Warehouse Model Based on Deep Reinforcement Learning

Jiaqing Chen ¹, Ming Jiang ^{2,*} and Guorong Chen ² ¹ School of Economics and Finance, Xi'an Jiaotong University, Xi'an 710061, China² School of Internet Economics and Business, Fujian University of Technology, Fuzhou 350118, China

* Correspondence: 19842158@fjut.edu.cn

Abstract

The multi-depot vehicle routing problem with soft time windows (MDVRPSTW) has long been a focus in both academic and industrial circles. This paper proposes a deep reinforcement learning framework designed to enhance the efficiency and quality of MDVRPSTW solutions, addressing the limitations of traditional heuristic algorithms in large-scale complex scenarios. The framework first transforms the mathematical model into a sequential decision-making problem through a Markov decision process, then extracts path selection strategies using an encoder–decoder architecture based on attention mechanisms and graph neural networks, and employs unsupervised reinforcement learning for model training. Test results on the Solomon benchmark dataset demonstrate that for small-scale problems ($N = 20$), our method reduces solving time by over 96% compared to comparative algorithms, with the objective value difference from the generalized variable neighborhood search (GVNS) being less than 9%. For medium-to-large scale problems ($N = 50/100$), our method achieves a 27.7 to 96.3 percent improvement over GVNS, maintaining stable solution times within 3 to 10 s. Compared to exact algorithms and meta-heuristic methods, our approach reduces computational costs by 2–3 orders of magnitude while demonstrating strong adaptability to variations in the number of depots and vehicles. In summary, this method significantly outperforms baseline models in both solution quality and computational efficiency, providing an efficient end-to-end solution for MDVRPSTW in complex scenarios.

Keywords: fresh food e-commerce; front-end warehouse; deep reinforcement learning; MDVRPSTW



Academic Editor: Wen-Chyuan Chiang

Received: 31 December 2025

Revised: 14 February 2026

Accepted: 26 February 2026

Published: 28 February 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and

conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Fresh food e-commerce is not only a key component but also a vital element of sustainable development. As high-frequency essential consumer goods, fresh products are characterized by short shelf life, susceptibility to spoilage, and heavy reliance on cold chains, which impose stringent logistics requirements. Xing Liu's research (2024) [1] proposed a multi-objective last-mile vehicle routing model for fresh food e-commerce, with three sustainability dimensions as independent objectives, optimizing last-mile routes simultaneously across these dimensions. Yan Huang's study (2025) [2] developed a hybrid forecasting model integrating Grey Relational Analysis (GRA), Wild Horse Optimization Algorithm (WHO), and Time Series Convolutional Network (TCN). This model addresses challenges in digital transformation in cold chain logistics demand forecasting, such as insufficient feature extraction and highly nonlinear data, thereby enhancing prediction accuracy.

The just-in-time delivery model based on front-end warehouse operations is gradually becoming the mainstream approach in fresh food e-commerce logistics and distribution. Scholars and industry professionals are increasingly focusing on optimizing existing logistics distribution models. Research on VRP (Vehicle Routing Problem) directly impacts the optimization results of front-end warehouse distribution patterns. In recent years, significant progress has been made in path planning studies across various fields. For instance, in the drone field, Xiaoduo Li proposed a hybrid heuristic algorithm (ALSA-RFC) with a robust feasibility test. This algorithm combines the advantages of adaptive large-scale neighborhood search and simulated annealing, enabling efficient handling of large-scale path planning problems [3]. Eryang Guo developed a two-stage evolutionary algorithm (TSC-PSODE) based on hybrid penalty strategies. This method not only provides an effective constraint processing mechanism but also achieves a good balance between exploration and exploitation by maintaining population diversity and accelerating convergence, effectively solving optimization problems with complex and dynamic constraints [4]. Phan Duc Hung proposed an adaptive ant colony optimization algorithm for unmanned vehicle path planning with time windows. This method is able to generate high-quality solutions in complex environments with random requests and tight time constraints [5].

Previous studies have achieved remarkable outcomes in combinatorial optimization domains, including vehicle routing and logistics scheduling. However, academic exploration of the MDVRPSTW (Multi-depot Vehicle Routing Problem with Soft Time Windows) remains limited. For instance, most existing research on MDVRPSTW employs heuristic algorithms [6–9], while studies that use deep reinforcement learning to solve the VRP often exclude time window constraints [10–12]. This paper proposes an innovative reinforcement learning approach to address these challenges in MDVRPSTW. First, we establish a learning decision mechanism through a Markov Decision Process (MDP) and detail decision-making elements such as vehicle path selection within the MDVRPSTW framework using a state–action–reward–policy architecture. The research team designed a multi-vehicle time-action mechanism based on an attention mechanism and a graph neural network encoder–decoder deep learning architecture, achieving network optimization through unsupervised reinforcement learning algorithms. Compared with traditional algorithms, this end-to-end solution framework can rapidly obtain high-quality solutions at various scales and effectively address the MDVRPSTW problem. Experimental results demonstrate that as problem complexity increases, the model exhibits enhanced performance in both solution quality and computational efficiency.

2. Literature Review

2.1. VRP Exact Algorithm and Heuristic Algorithm

Previous studies on VRP have primarily employed three approaches: exact algorithms, approximate algorithms, and heuristic algorithms. Exact algorithms are designed to find optimal solutions, though they perform well for small-scale VRP cases but struggle with large-scale ones. For instance, Najib Errami’s VRPSolverEasy can solve multiple VRP variants optimally and obtain suboptimal solutions within time constraints, though its computational time becomes prohibitively long when applied to over 100 customer nodes [13]. Meta-heuristic algorithms have gained widespread adoption due to their superior performance. Malek Masmoudi’s the Generalized Variable Neighborhood Search (GVNS) algorithm, which is specifically designed for VRP problems with time windows, features dynamic penalty mechanisms, multi-neighborhood structures, and systematic parameter tuning, and demonstrates high efficiency, robustness, and performance advantages in both solution quality and computational time [14]. Brenner Humberto Ojeda Rios developed three methods for the stochastic capacity-constrained multi-depot VRP (SCMVRPPD,

Stochastic Capacitated Multi-depot Vehicle Routing Problem with Pickup and Delivery): tabu search (TS) heuristic, generalized variable neighborhood search (GVNS), and iterative local search (ILS-VND). Their research revealed that ILS-VND consistently achieved the best solution quality [15]. Shifeng Chen proposed a hybrid approach to solve the VRP problem with dynamic requests. The method first constructs an initial solution using GRASP, then explores and refines it with VND, and finally demonstrates its competitive advantage through two benchmark cases of dynamic pickup and dynamic delivery [16].

2.2. VRP Deep Reinforcement Learning Algorithm

Researchers have employed deep reinforcement learning in conjunction with neural networks to address the vehicle routing issue, taking advantage of these techniques' strong capacity to represent and learn from data. In the VRP, selecting decision variables and making sequential decisions in discrete decision spaces are key steps, and RL is well-suited for the latter due to its inherent advantages. The functions of DRL have natural similarities with this sequential decision-making process; the "offline training" and "online decision-making" features of DRL enable real-time online solving of the VRP. Qingshu Guan proposed a dynamic embedding-based DRL (DE-DRL) for heterogeneous capacity vehicle routing (HCVRP), which utilizes an innovative encoder–updater–decoder (EUD) framework. Empirical results demonstrate that DE-DRL consistently outperforms heuristic methods and other DRL approaches [17]. VRP-STC, which is a complex extension of the classical VRP paradigm, incorporates stochastic elements into travel cost calculations. In this problem, the vehicles face not only capacity constraints but also varying travel costs between each node pair, which are characterized by random variables. Hao Cai developed a Graph Attention Network (GAT)-AM model combining GAT and AM mechanisms. The GAT-AM model adopts an encoder–decoder architecture and employs deep reinforcement learning algorithms to solve VRP-STC. Empirical findings indicate that the model achieves better solution quality as the problem complexity increases [18]. Yujun Wang also investigated the heterogeneous vehicle routing problem with service time constraints (HVRP-STC) and modeled it as a Markov decision process with service time constraints. He proposed a novel deep reinforcement learning-based model called TDRL (Weighted Deep Reinforcement Learning). Empirical results demonstrated that TDRL consistently outperformed the most advanced DRL methods at that time [19].

The types of VRP variants studied in the aforementioned literature are shown in Table 1.

Table 1. Literature comparison.

Document	Multi-Depot	Capacity Requirements	Window Requirement	Dynamic Request
[13]	✓ *	✓	✓	✓
[14]	✓	✓	✓	
[15]				
[16]				
[17]				
[18]				
[19]				
This article	✓	✓	✓	

* The √ symbol indicates that the research question in the corresponding literature involves a specific variant of VRP.

3. Problem Definition

This study centers on the central aspect of the front warehouse mode distribution optimization problem in relation to the multi-depot vehicle routing problem with soft

time windows, drawing upon the research conducted by previous scholars. We propose a specific mathematical model of MDVRPSTW and model it as a representation of the relevant elements of a Markov process (MDP) in this section.

3.1. Problem Description

The optimization problem based on MDVRPSTW can be formulated as follows: We model the road network as a connected graph $G = (V, E)$, where V represents the set of all nodes and E represents the set of all edges. Specifically, the distribution area contains M distribution centers, each of which is equipped with K_m delivery vehicles with a rated cargo capacity of Q units. The area is served by N customers, each with demand quantity q_i and known time windows. Vehicles arriving before the E_i or after the L_i fail to meet customer demand within their designated time windows and thus incur penalty costs for premature or delayed arrivals. Vehicles arriving within the time window $[E_i, L_i]$ for customer i incur no penalty costs. All penalty costs are uniformly converted into unit time costs and incorporated into the edge weight of E .

Given the locations x_i of each customer node, the time windows $[E_i, L_i]$, the demand quantities the q_i , the locations of depots, the rated capacity Q of each vehicle, and the total number of vehicles K , optimal routes should be designed to minimize the total delivery time cost.

The definition of mathematical symbols is shown in Table 2.

Table 2. Variable Definitions.

Symbol	Definition	Remarks
Z	total distribution cost	$\begin{aligned} \min Z &= \sum_{k=1}^K \sum_{i=0}^N \sum_{j=0}^N d_{ij} x_{ijk} \\ &+ \alpha \sum_{i=1}^N \max(E_i - t_{ik}, 0) \\ &+ \beta \sum_{i=1}^N \max(t_{ik} - L_i, 0) \end{aligned}$
D_{ij}	The travel cost from node i to node j	-
M	Number of vehicle lots (front warehouses)	2 or 3
K	Total number of vehicles	$M \times K_m$
K_m	Number of configured vehicles in the m -th depot	2 or 3
N	Number of client nodes	20 or 50 or 100
t_{ik}	The time when the k -th vehicle arrives at the i -th customer node	Start timing from the depot
x_{ijk}	Whether the k -th vehicle is from i to j	0 = false, 1 = true
x_i, y_i	Geographical coordinates of the i -th customer node	coordinate normalization
q_i	Demand of the i -th customer node	randomly distributed experiment
Q	rated load capacity of vehicle	Q is adjusted according to the number of customer nodes
E_i	Service start time for the i -th client node	Determine the random range based on the number of client nodes
L_i	Service end time for the i -th client node	
α	early arrival penalty factor	[0.2, 0.4]
β	late time penalty factor	[0.6, 0.8]

3.2. Constraints

Based on the operational rules of front-end warehouse distribution in fresh food e-commerce, we established 11 rigorous mathematical constraints, with their expressions and physical meanings described and shown in Table 3.

Table 3. Mathematical Constraints and Their Physical Meanings.

Constraint	Formula	Description
vehicle load	$\sum_{i=1}^N q_i x_{ijk} \leq Q, \forall k \in K$	The total demand of all customer nodes served by a single vehicle should not exceed its rated load capacity.
Unique client node access	$\sum_{k=1}^K \sum_{j=0}^N x_{ijk} = 1, \forall i \in N$	Each client node can only be accessed by one vehicle to ensure service uniqueness
conservation of flow of vehicles	$\sum_{i=0}^N x_{ijk} = \sum_{j=0}^N x_{jik}, \forall k \in K, \forall i \in N$	For any vehicle, the number of times it leaves a node equals the number of times it arrives at that node, avoiding path interruption.
vehicle deployment frequency	$\sum_{i=0}^N x_{0ik} \leq 1, \forall k \in K$	Each vehicle can only be dispatched once from the depot, with no repeated empty trips.
Number of vehicles departing from the depot	$\sum_{k=1}^{K_m} x_{0ik} \leq K_m, \forall m \in M$	The number of vehicles dispatched from each depot does not exceed the total number of vehicles allocated to it.
vehicle path closure	$\sum_{i=1}^N x_{i0k} = \sum_{i=0}^N x_{0ik}, \forall k \in K$	The vehicle must return to its original depot after completing the delivery, forming a closed path.
Vehicle arrival time updated	$t_{jk} = t_{ik} + x_{ijk}, \forall i, j \in N, \forall k \in K$	The time for a vehicle to reach the next customer node equals the sum of the current node arrival time and the travel time between the two nodes, with the travel time assumed to be 1 unit time.
nonnegative decision variable	$x_{ijk} \geq 0, \forall i, j \in N, \forall k \in K$	
Decision variable	$x_{ijk} \in 0, 1, \forall i, j \in N, \forall k \in K$	$x_{ijk} = 1$ indicates that the k-th vehicle travels from node i to node j, while $x_{ijk} = 0$ denotes the opposite scenario.
The arrival time is non-negative	$t_{ik} \geq 0, \forall i \in N, \forall k \in K$	The time when the vehicle arrives at the customer node is a non-negative value.
depot Number	$0 \in M, i, j \in N \cup M$	Set the depot number to 0 and the customer node number to 1 to N and include them in the unified path planning system.

3.3. Overview of MDP

By modeling the route solution of the MDVRPSTW as an MDP, this paper transforms the problem into a sequential decision-making problem. The specific process is shown in Figure 1. The tuple $M = \{S, A, \tau, r\}$ serves as the primary definition of the MDP. This section outlines the state space, action space, state transition, and reward function, respectively.

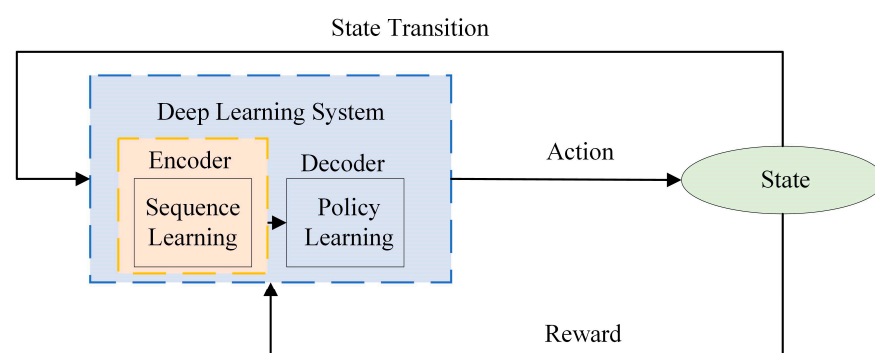


Figure 1. Detailed process of Markov decision process. The blue dashed box represents the Deep Learning System, the core module of the framework. The orange dashed box inside the blue box represents the Encoder, which is responsible for sequence learning. The green oval represents the State of the environment. The arrow from the Encoder to the Decoder represents the flow of encoded latent representations. The arrow labeled *Action* represents the decision output from the Decoder to the environment. The arrow labeled *Reward* represents the feedback signal from the environment to the Decoder for policy optimization. The arrow from the State back to the Encoder represents the closed-loop input of the new environment state for the next iteration.

The state space consists of two components, the global static state S_g and the vehicle dynamic states S_d , and these two components together form the complete decision state.

Global Static State S_g : The inherent invariant parameters of the problem, including geographic coordinates (x_i, y_i) of all nodes, the demand q_i of each customer node, customer service time windows $[E_i, L_i]$, vehicle rated load Q , number of depots M , vehicle allocation per depot K_m , total vehicle count K , the travel cost D_{ij} between each pair of nodes, early arrival penalty factor α , and late arrival penalty factor β .

Dynamic State S_d : The parameters updated in real-time during the decision-making process, which represent the decision states of individual vehicles. Each vehicle's dynamic state is denoted as $S_{dk} = \{(x_0, d_0), (x_1, d_1), \dots, (x_t, d_t)\}$, where x_t denotes the customer node/lot node that vehicle k is ready to access at time t , and d_t represents the remaining load after vehicle k accesses x_t . The dynamic state must also include a global node mask at time t . Note that if all customer nodes in the global node mask are prohibited and the current action ends, the system will first force the next action to return all vehicles to the depots before terminating the decision-making process.

The overall state space $S = S_g \cup \{S_{d1}, S_{d2}, \dots, S_{dK}\}$ represents the set of global static parameters and all vehicle dynamic states, where every state is valid.

Action Space: The core decision-making objective of the action space is to assign the next node to visit for each vehicle currently in a valid decision state, enabling synchronized path decisions among multiple vehicles. The action form of a single decision is defined as $a_t = \{x_{t1}, x_{t2}, \dots, x_{tK}\}$, where K denotes the total number of vehicles, and x_{tk} represents the next node to visit assigned to the k -th vehicle at step t (including customer nodes and depot nodes; if no movement occurs, the vehicle remains at the node from time $t - 1$). To ensure the legitimacy of action space A , a vehicle capacity mask is computed based on d_t before node selection (nodes from time $t - 1$ are always available). The global mask is then replicated, AND-ed with the vehicle capacity mask, and the current node is set as accessible before calculating action probabilities. Once the i -th vehicle selects its node, the global node mask is immediately updated.

State transition: The state transition function $\tau: S \times A \rightarrow S$ is defined as the process where, under the legal state st at time t , executing the synchronous decision action at updates the decision state from st to the legal state s_{t+1} at time $t + 1$. The core logic of state transition involves independent updates of multiple vehicle states, unified global constraint verification, real-time mask synchronization, and precise sequential progression. All update operations strictly adhere to the constraints of MDVRPSTW (vehicle load capacity, unique node access, number of vehicles departing from the depot, etc.). The specific update rules are as follows: 1. The global static state S_g remains unchanged and contains inherent parameters such as node coordinates, demand quantities, time windows, and penalty factors; these parameters remain constant throughout the decision-making process and serve solely as constraints for state transitions. 2. Vehicle Dynamic State S_d : For each vehicle k ($k = 1, 2, \dots, K$) which is synchronized independently, its dynamic state S_{dk} is updated based on the next node x_{tk} assigned in action at (where t is the t -th step assigned to the k -th vehicle). The core update rule is as follows: When updating the access-ready node, replace "node x_t ready to access at time t in S_{d+1} " with "node x_{t+1} ready to access at time $t + 1$ ", i.e., update S_{d+1} 's core structure from $\{(x_0, d_0), \dots, (x_t, d_t)\}$ to $\{(x_0, d_0), \dots, (x_t, d_t), (x_{t+1}, d_{t+1})\}$. Remaining load updates are as follows: $d_{t+1} = d_t - q_{t+1}$ (where q_{t+1} is the customer demand for x_{t+1} ; if x_{t+1} is a time node at step t , $d_{t+1} = d_t$; if a depot node, $d_{t+1} = Q$). Global node mask updates are as follows: Strictly follow the rule "update immediately after the i -th vehicle selects its node"—during action execution, mark x_{tk} (customer node) as "allocated" immediately after the completion and validation of the assignment of x_{tk} to each vehicle, ensuring subsequent vehicle node selection uses the updated mask to

prevent redundant allocation. Time updates are as follows: The global decision-making time uniformly progresses from t to $t + 1$, which guarantees complete synchronization of all vehicles' decision timing to avoid constraint conflicts caused by temporal asynchrony.

Reward function: The goal of the MDVRPSTW is to reduce the total time needed to complete the delivery task. This study defines the reward function as the negative value of the objective function $R = -\sum_{i=1}^m \sum_{t=1}^T r_t^m = -\sum_{i=1}^m \sum_{t=1}^T (d_t^m + \varepsilon_t^m)$. The objective is to minimize the sum of vehicle driving distance and penalty time.

4. Theory

This section presents a deep reinforcement learning model based on an attention mechanism and Graph Neural Network (GNN) for the Markov Decision Process (MDP) model defined in Part 3 of MDVRPSTW. The model adopts an end-to-end encoder–decoder architecture, employing unsupervised reinforcement learning to achieve policy representation that precisely meets path sequence decision-making requirements under multi-vehicle, multi-vehicle, and soft time window constraints. We supplement the interpretability analysis from three dimensions: the physical significance of feature selection, the logical correlation of feature fusion, and the generation mechanism of path decisions. Figures 2 and 3 respectively demonstrate the decision execution process and network architecture of the model for the MDVRPSTW problem. All stages of the model's encoding, decoding, and training are centered around the core objective of minimizing total delivery time cost in MDVRPSTW. Key constraints such as global node masks, vehicle capacity masks, and multi-vehicle dynamic state updates are integrated to ensure the legitimacy and interpretability of the output decisions.

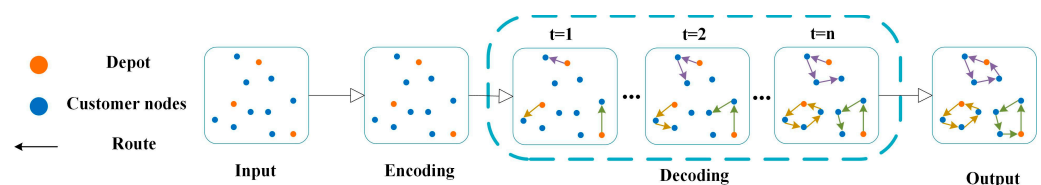


Figure 2. Process of solving MDVRPSTW. The arrows in the figure with different colors, represent the action sequences of different vehicles.

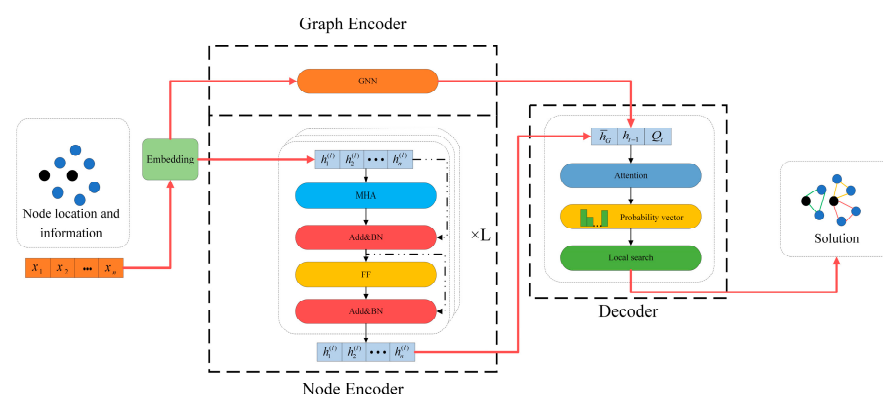


Figure 3. Specific architecture of neural network. The blue and black spheres represent the nodes in the distribution network, where black spheres denote depots and blue spheres denote customer nodes. The outermost dashed box labeled *Graph Encoder* represents the overall graph encoding module. The inner dashed box labeled *Node Encoder* represents the sub-module responsible for encoding node features. The dashed box labeled *Decoder* represents the decision-making module that generates solutions. Black arrows represent the flow of data and feature information through the network layers. Red arrows represent the flow of key intermediate representations (node embeddings) between the encoder and decoder, as well as the final output of the solution.

4.1. Model Overview

The framework of this model follows a “feature selection–feature fusion–constraint filtering–probability decision–state update” sequence. Feature selection: The encoder selectively retains features strongly relevant to the MDVRPSTW’s optimization objective (geographical coordinates, demand volume, time windows, driving costs, vehicle load capacity) while eliminating irrelevant redundant features to ensure rational feature selection. Feature fusion: Through multi-head attention mechanisms, the system captures feature correlations between customer nodes, while Graph Neural Networks (GNNs) identify topological relationships between vehicle depots and customers. The integrated features directly reflect “delivery priority, route adjacency, and resource compatibility”. Constraint filtering: The decoder pre-filter invalid decisions using capacity masks and global node masks, enforcing practical delivery rules such as “prohibiting vehicle overloading and avoiding duplicate customer service”. Probability decision: The decoder outputs node selection probability distributions, where weights directly correspond to the model’s assigned delivery priority for each node; higher probabilities indicate higher service priority under current conditions. State update: Decision-based state transitions strictly adhere to Markov Decision Process (MDP) rules, implementing real-time updates of the remaining vehicle load capacity, served nodes, and delivery time after the vehicle completes service at each node.

The model defines the random selection strategy P as the probability distribution output by the policy network π_θ over the action space A . Specifically, for the current state s_t , the policy network generates probability distributions for multi-vehicle selection at each node. The random strategy P samples valid actions from these distributions, ensuring the exploration of a more optimal path decision space during training. The generation of probability distributions represents the core manifestation of the model’s interpretability. The random selection strategy P is defined as follows:

$$p(s_\tau | s_0) = \sum_{t=0}^{\tau-1} \pi_\theta(a_t | s_t) p(s_{t+1} | s_t, a_t) \quad (1)$$

4.2. Encoder Architecture

This subsection will provide a comprehensive introduction to the distinct structures of node and graph encoders. The node encoder is employed to acquire the attributes of each node, while the graph encoder is utilized to ascertain the possible connections among nodes. To begin with, incorporate feature $X_i = [x_i, d_i, e_i, l_i]$ of all customer nodes into the plane coordinate system, where x_i is the location information of the customer node and distribution center, d_i is the distribution demand of the customer node, and e_i and l_i are the time window information. We begin the customer node embeddings by calculating them using learnable linear projections W_0 and b_0 :

$$h_i^{(0)} = W_0 \bullet \text{Concat}(x_i, d_i, e_i, l_i) + b_0 \quad (2)$$

The initial node embeddings $h_i^{(0)}$ are then fed into the node encoder and graph encoder.

4.2.1. Node Encoder

In layer l , the node encoder is composed of multi-head attention, with d_k denoting the dimension of each query/key, d_v representing the dimension of value, and h indicating the number of heads in the attention head. In particular, the MHA of the l -th layer encoder

initially computes a for every attention head, subsequently merging these attention heads. The following are the steps to take:

$$Q_i^{c,l} = W_i^{Q,c,l} h_i^l, K_i^{c,l} = W_i^{K,c,l} h_i^l, V_i^{c,l} = W_i^{V,c,l} h_i^l \quad (3)$$

$$head_i^{c,l} = \text{soft max}\left(\frac{Q_i^{c,l} (K_i^{c,l})^\top}{\sqrt{d_k}}\right) V_i^{c,l} \quad (4)$$

$$MHA(h_i^l) = \text{Concat}(head_i^{1,l}, head_i^{2,l}, \dots, head_i^{C,l}) W_i^{O,l} \quad (5)$$

Here $W_i^{Q,c,l} \in \mathbb{R}^{h \times d \times d_k}$, $W_i^{K,c,l} \in \mathbb{R}^{h \times d \times d_k}$, $W_i^{V,c,l} \in \mathbb{R}^{h \times d \times d_v}$ and $W_i^{O,l} \in \mathbb{R}^{d \times d}$ are learnable parameters in the MHA layer. On each attention head, the correlation between them is calculated, and finally, the attention features obtained by each attention head are spliced to obtain better feature representation. After that, the feedforward neural network, residual connection and batch normalization are used to process the multi-head attention output of the l-th layer with the following formulas:

$$m_i^l = \text{BN}(h_i^l + MHA^l(h_i^l)) \quad (6)$$

$$h_i^{l+1} = \text{BN}(M_i^l + FF(M_i^l)) \quad (7)$$

Ultimately, by utilizing the encoder node embedding $h_i^{(0)}$, the problem instance's node encoding is acquired.

4.2.2. Graph Encoder

A graph neural network (GNN) is employed to capture the potential relationships between nodes in a convenient and dynamic manner, thereby representing the graph structure of the problem. The following are the definitions for each layer:

$$GNN^l(X_i^{l-1}) = \lambda X_i^{l-1} \Theta + (1 - \lambda) \Phi_\theta(X_i^{l-1} / |N(i)|) \quad (8)$$

$$H_i^c = GNN^l(h_i^c) \quad (9)$$

The graph's edge weight can be adjusted using the trainable parameter λ . Θ is a parameter that can be trained, $N(i)$ is the set of nodes that are next to each other, and Φ_θ is a function that aggregates contextual information. Finally, the graph embedding H_c^t is obtained through the graph encoder.

4.3. Decoder Architecture

The decoding process of the decoder begins with the input of the encoded depot and node features and graph embeddings, followed by the generation of the probability distribution of all nodes and depots through the attention mechanism. Masking is employed to conceal depots and nodes that fail to satisfy the relevant constraints. Ultimately, node selection is accomplished by employing various search techniques, for example, a greedy search that selects the node with the highest probability and a sampling search based on a probability distribution. The decoder uses the encoder's embedding, the output of the previous time step, the time of the current time step, and the vehicle's remaining load capacity to select the next node that needs service at each time step. The decoding process continues until all customers are served. The ultimate goal of decoding is to generate an optimal routing solution by learning effective path selection strategies. Initially, establish

the contextual information h_t^c , comprising the graph embedding H_{tc} , the preceding node traversed by the chosen vehicle k , and the residual capacity $D_{m,t}$ of the vehicle.

$$h_t^c = \begin{cases} [H_t^c, h_{\pi_{t-1}}, D_{m,t}], t > 1 \\ [H_t^c, h_{\pi_0}, D_{m,t}], t = 1 \end{cases} \quad (10)$$

Then, we generate probability distribution based on the relationship between contextual information H_t^c and embedded node h_N , and select probability vectors through single-headed attention output nodes:

$$u_t = C \times \tanh\left(\frac{(W_Q H_t^c) \cdot (W_K h_N)}{\sqrt{d_K}}\right) \quad (11)$$

The decoder executes a mask operation on the client points by utilizing $C = 10$ to truncate u_i within $[-C, C]$, where W_Q and W_K are trainable parameters. Utilize the e function for normalization in order to derive the selection probability $p_{i,t}$ for every node; subsequently,

$$p_t = \text{soft max}(u_t) = \frac{e^{u_t}}{\sum_j e^{u_j}} \quad (12)$$

At the training stage, we adopt the sampling decoding method based on the output probability $p_{i,t}$ of the decoder. In the test phase, the model adopts the greedy decoding method that selects the node with the maximum probability value $p_{i,t}$.

4.4. Training Strategy

Policy gradient is used here to train the model. The target $L(\theta | s)$ is the expected reward, which will be evaluated based on parameter θ :

$$\nabla_{\theta} L(\theta | s) \approx E_{p_{\theta}(a|s)}[(R(a | s) - R^{BL}(s)) \nabla \log p_{\theta}(a | s)] \quad (13)$$

In this training process, two network representations are employed: (1) Policy network $R(a | s)$, which analyzes the probability distribution $p_{i,t}$ to determine the overall cost of the sample; (2) Baseline network $R^{BL}(s)$, which evaluates the training process by identifying the positive and negative deviations from the baseline $R^{BL}(s)$ and eliminating disparities in training. The average target $\nabla_{\theta} L(\theta | s)$ for each training cycle will be obtained through the sampling strategy, and the parameter θ is updated using the Adam optimizer.

The algorithm flow is shown in Algorithm 1.

Algorithm 1: Deep Reinforcement Learning Algorithm

Input:	Batch size B ; Data size D ; Number of epochs N ; Maximum training steps Γ ; Initial parameters θ for policy network π_{θ} ; Initial parameters Φ for baseline network v_{Φ} ;
Output:	The optimal parameters θ, Φ
1	$T \leftarrow D/B$
2	for epoch = 1, 2, ..., N do
3	for $t = 1, 2, \dots, T$ do
4	Retrieve batch $b = B_t$
5	for $i = 0, 1, \dots, \Gamma$ do
6	Select an action $a_{t,b} \sim \pi_{\theta}(a_{t,b} s_{t,b})$;

Algorithm 1: *Cont.*

```

7           Calculate rewards  $r_{t,b}$  and update status  $s_{t+1,b}$ ;
8           end
9            $R_b = -\max(\sum_{t=0}^{\Gamma} r_{t,b})$ 
10          GreedyRollout with baseline  $V_{\phi}$  and Compute  $R_b^{BL}$ 
11           $d_{\theta} \leftarrow \frac{1}{B} \sum_{j=1}^B [(R_b(a | s) - R_b^{BL}(s)) \nabla \log p_{\theta}(a | s)]$ 
12           $\theta \leftarrow Adam(\theta, d_{\theta})$ 
13          If PairedTTest ( $\pi_{\theta}, \nu_{\phi}$ ) < 5%
14               $\phi \leftarrow \phi$ 
15          end
16          end

```

5. Experiment

This paper generates the training data through random distribution, compares the proposed model with the baseline model using Solomon's (1987) [20] dataset, and validates the proposed approach by evaluating the computational efficiency and model transferability of both approaches. Finally, parameter sensitivity experiments are conducted to analyze the impact of various parameters on learning performance.

5.1. Experimental Setup

Within this section, we elucidate the experimental configuration and the methodology employed to generate the data. The warehouses and customers are evenly distributed, and customer demand is evenly distributed among them. The distribution of time windows follows a uniform pattern across 20 client nodes. The time windows are evenly distributed between 50 and 100 client nodes. Each vehicle has a load capacity of 60, 150, and 300 nodes, respectively, when measured at 20, 50, and 100 nodes. Table 4 provides a comprehensive description of the fundamental configurations for generating data. The early arrival penalty factor α and late arrival penalty factor β of the time window are randomly generated from [0.2, 0.4] and [0.6, 0.8], respectively. Here, we take into account two depots and three depots, each accommodating a maximum of two and three vehicles, respectively.

Table 4. Basic settings for data generation.

Problem Size	Number of Customer Nodes	Node Coordinates	Customer Point Demand	Maximum Vehicle Load Capacity	Time Window Distribution
MDVRPSTW20	20	$x_i \in [0, 1]^2$	$d_i \in [1, 9]$	Q = 60	[0, 20]
MDVRPSTW50	50	$x_i \in [0, 1]^2$	$d_i \in [1, 9]$	Q = 150	[0, 50]
MDVRPSTW100	100	$x_i \in [0, 1]^2$	$d_i \in [1, 9]$	Q = 300	[0, 100]

The training instances are generated at random according to the settings, with an iteration size of 1,280,000 and a batch size of 1024, for a total of 100 epochs at each problem size. The Adam optimizer is employed to train the policy parameters while the learning rate is fixed at 1×10^{-4} . Generally, the more time spent training, the more successful the performance will be. If the performance improvement is not obvious after a large number of iterative trainings, the iterations can be stopped before convergence, which can greatly save training time, and the learning effect is very competitive. The tactic of halting early is established at this location for a duration of 20 rounds. The platforms utilized for conducting all experiments here include NVIDIA GeForce RTX 3090 (24G video memory)

and Xeon (R) Platinum 8255C. The experiments were carried out using PyTorch 2.7.1. The hidden layer dimension is set to 128, there are 8 attention heads, and the model has 3 layers.

5.2. Baseline

This study conducts comparative analysis between two representative baseline algorithms. The selection of these algorithms adheres to the principle of “covering different solution paradigms while aligning with the core characteristics of the problem”: one is an exact algorithm for VRP-type problems, used to verify the optimal boundary of solutions in small-scale scenarios; the other is a classical meta-heuristic algorithm designed for time-windowed VRP (VRPTW), used to validate the model’s generalization and superiority in complex scenarios. All comparative experiments are based on the classic VRPTW dataset by Solomon (1987) [20] and undergo standardized preprocessing tailored to the MDVRPSTW problem characteristics, ensuring fairness and validity of the comparison.

5.2.1. Exact Algorithm: VRPSolverEasy [13]

Core algorithm features: The algorithm operates without heuristic rules and employs an enumeration-and-pruning approach to traverse the solution space. Given sufficient computational time, it can rigorously compute the global optimal solution. Even under time constraints, the adaptive pruning strategy efficiently filters out high-quality suboptimal solutions, balancing computational accuracy with time efficiency.

Adaptation to the research problem: VRPSolverEasy is natively designed for single-vehicle depot VRP problems. To address the multi-depot (MDVRPSTW) characteristics of this study, the “subproblem decomposition” strategy is adopted. The multi-depot problem is decomposed into multiple independent single-depot VRPTW subproblems, each corresponding to a depot’s customer set while maintaining the original constraints (e.g., time windows, demand, and load capacity) for customer nodes. The synchronization delivery constraint is implemented by adding a “cross-depot delivery rhythm penalty term” to the objective function, ensuring the algorithm’s solution logic aligns with the study’s problem definition.

Evaluation parameter settings: To match the experimental environment of this study, the maximum solution time threshold for VRPSolverEasy was set as follows: 300 s for small-scale problems ($N = 20$), 1800 s for medium-scale problems ($N = 50$), and 3600 s for large-scale problems ($N = 100$). Other parameters were maintained at the algorithm’s default configuration to maximize its solution performance.

Core objectives: The study evaluates the DRL model’s solution quality and deviation from global optimal solutions in small-scale scenarios using VRPSolverEasy’s results, while quantifying the time cost of the exact algorithm in medium-to-large-scale scenarios to demonstrate the DRL model’s efficiency advantage.

5.2.2. Meta-Heuristic Algorithm: Generalized Variable Neighborhood Search (GVNS) [14]

Core algorithm features: This algorithm is specifically designed for the VRPTW with synchronization constraints, and it demonstrates strong local optimization capability and convergence efficiency. Its core neighborhood structure incorporates four operations including intra-path node exchange and inter-path node redistribution, effectively avoiding local optima. In bike parking VRPTW scenarios, it outperforms traditional meta-heuristic algorithms such as Simulated Annealing (SA) and Genetic Algorithm (GA) in both solution quality and computational efficiency.

Adaptation to the research problem: Given that GVNS is originally designed for single-vehicle scenarios, we implement a two-phase adaptation strategy (allocate first, optimize later) for the multi-vehicle scenario (MDVRPSTW). Phase 1: Pre-allocation. Using K-means clustering, all customer nodes are assigned to designated vehicle depots (with cluster

centers corresponding to depot locations), ensuring the total customer demand of each depot satisfies the vehicle capacity constraints. Phase 2: Optimization. For each depot's customer subset, the GVNS algorithm is independently executed to compute optimal routes. The synchronization constraint is achieved by adjusting the time window relaxation coefficient, ensuring synchronized delivery schedules across different depots.

Evaluation parameter settings: Based on Malek Masmoudi's original study [14], the GVNS algorithm was configured with 1,280,000 iterations (consistent with the DRL model training iterations in this study). The neighborhood structure incorporated four operations: node swapping (N_1), node insertion (N_2), path splitting (N_3), and path merging (N_4). The perturbation intensity was dynamically adjusted during iterations, with strong perturbations in early stages to ensure global exploration and weak perturbations in later stages to enhance local optimization. The maximum iteration stagnation threshold was set at 20,000 steps to trigger the early stopping mechanism.

Core comparison: As the benchmark algorithm for VRPTW in single-vehicle scenarios, GVNS demonstrates its solution quality in multi-vehicle scenarios, validating the DRL model's superiority in cross-vehicle generalization and complex constraint adaptation. The comparative analysis of solution time and quality across problem scales further highlights DRL's efficiency and stability in medium-to-large-scale scenarios, particularly its time advantage during exponential solution space expansion.

5.2.3. Solomon (1987) [20] Dataset Preprocessing

To ensure consistency in comparative experiments, the Solomon dataset underwent the following preprocessing steps: 1. Sample Screening: The R1 and C1 series (covering scenarios with varying customer distribution densities) were filtered, excluding samples with demand exceeding the rated load capacity of vehicles in this study. 2. Coordinate Normalization: The node coordinates of the original dataset were normalized to a 2D square space $[0, 1] \times [0, 1]$ (consistent with the input features of the DRL model). 3. Multi-Depot Expansion: A total of 2 or 3 depot locations were randomly generated within the normalized space to ensure uniform distribution. 4. Constraint Adaptation: The departure time windows for vehicles in each depot were unified to a single time range to align with synchronous delivery constraints, with early arrival/late arrival penalty factors consistent with the DRL model settings ($\alpha \in [0.2, 0.4]$, $\beta \in [0.6, 0.8]$).

The preprocessed data were categorized into three scales, small scale ($N = 20$), medium scale ($N = 50$), and large scale ($N = 100$), which perfectly matched the experimental scale of this study.

6. Results

This section compares the deep reinforcement learning model proposed in the text with the baseline model on the MDVRPSTW problem in terms of solution quality and computational efficiency. Figure 4 presents the model's solution for a scenario with 2 depots, 2 vehicles per lot, and 20 customers. Table 5 presents the test results across three problem scales with varying numbers of depots and vehicles per depot. Based on these findings, the following conclusions can be drawn.

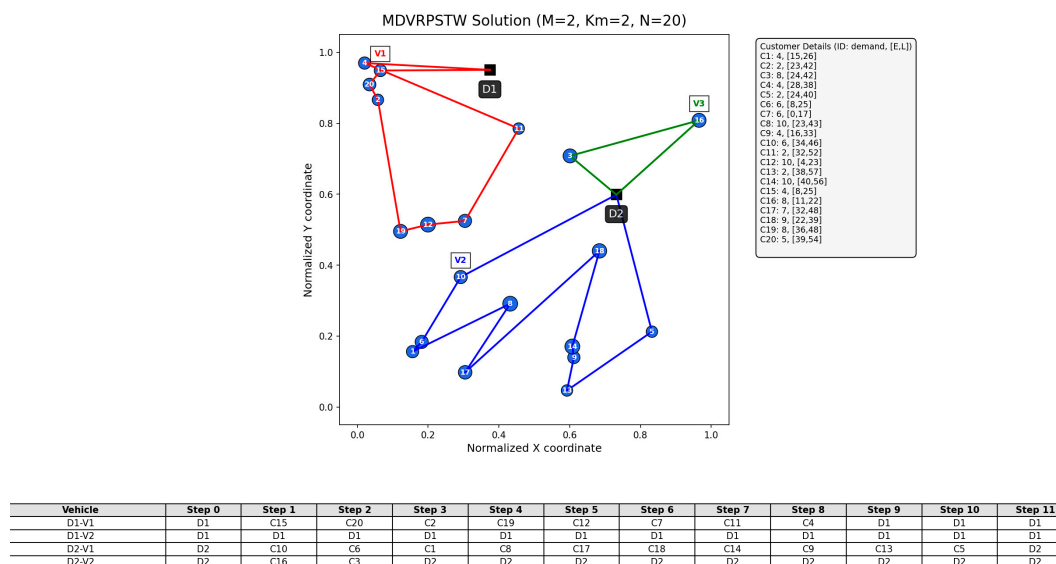


Figure 4. Example of results for 2 depots, 2 vehicles per depot, and 20 customer nodes. Red lines represent the route of the 1st vehicle from depot D1. Blue lines represent the route of the 1st vehicle from depot D2. Green lines represent the route of the 2nd vehicle from depot D2. Black squares (D1, D2) represent the depots (distribution centers). Red dots represent the customer nodes visited by the 1st vehicle from D1. Blue dots represent the customer nodes visited by the 1st vehicle from D2. Green dots represent the customer nodes visited by the 2nd vehicle from D2. The table below shows the actions selected by the four vehicles at each time step.

Table 5. Comparison of the effects of solving MDVRPSTW calculation examples.

Depot	Vehicle	Method	N = 20		N = 50		N = 100	
			Obj	Time/s	Obj	Time/s	Obj	Time/s
2	2	VRPSolverEasy	60.12	89.56	289.45	1789.23	876.54	Timeout (3600 s)
		GVNS	74.25	98.76	214.89	456.89	488.76	5689.34
		DRL	88.98	3.45	175.67	3.12	261.89	4.58
	3	VRPSolverEasy	57.89	95.34	278.67	1890.56	865.43	Timeout (3600 s)
		GVNS	69.87	105.43	207.56	525.78	477.89	6123.45
		DRL	74.89	4.23	142.56	3.58	185.78	4.27
3	2	VRPSolverEasy	58.98	92.67	290.78	1901.34	888.67	Timeout (3600 s)
		GVNS	71.98	110.89	204.67	610.98	468.90	7201.56
		DRL	76.89	3.12	135.89	7.89	210.67	4.12
	3	VRPSolverEasy	56.78	98.90	277.89	2010.67	864.32	Timeout (3600 s)
		GVNS	68.76	118.56	194.56	688.76	457.89	7890.12
		DRL	69.87	3.25	139.89	9.78	228.90	4.20

Objective metrics: DRL outperforms VRPSolverEasy by 32.9% at $N = 20$ and achieves an objective value 27.7% lower than GVNS at $N = 50$ and 96.3% lower than GVNS at $N = 100$, demonstrating a “scale-dependent advantage” pattern. Time metrics: DRL’s average execution time (4.63 s) across all scenarios is merely 0.19% of GVNS (2468.06 s) and 0.67% of VRPSolverEasy (696.03 s, excluding timeouts), showing exceptional time stability (coefficient of variation: 18.7%). Configuration adaptability: DRL’s objective response to configuration changes (−20.15%) is 4.77 times stronger than VRPSolverEasy’s (−4.22%), making it the most adaptable to multi-depot and multi-vehicle configurations.

6.1. Generalization Experiments

It is important to take into account that the quantity of customers and the weight of the vehicle are not predetermined during the actual delivery process. For instance, when attempting to resolve the issue of 100 customer points, certain customer points may not

need to be delivered on the same day, or the vehicle may need to be replaced and its weight altered. This model is adept at addressing this issue. This model, which has been trained, is utilized to address issues of varying customer sizes, and any customer point can still be addressed swiftly. The trained models with 20, 50, and 100 customer sizes are adjusted according to the case size, and the results are displayed in Tables 6–8.

Table 6. Generalization experiment at 20-node scale.

Depot	Vehicle	N = 16	N = 17	N = 18	N = 19	N = 20
2	2	77.477	72.625	79.333	89.006	89.594
	3	71.322	73.865	74.987	73.354	75.488
3	2	80.704	79.307	75.809	76.798	77.331
	3	77.732	78.781	78.781	74.213	70.43

Table 7. Generalization experiment at 50-node scale.

Depot	Vehicle	N = 46	N = 47	N = 48	N = 49	N = 50
2	2	173.552	173.291	158.226	154.212	176.51
	3	143.702	143.335	130.600	149.620	149.067
3	2	132.082	119.198	124.564	119.198	143.143
	3	134.751	125.073	140.793	167.620	140.642

Table 8. Generalization experiment at 100-node scale.

Depot	Vehicle	N = 96	N = 97	N = 98	N = 99	N = 100
2	2	217.180	227.685	238.859	259.606	262.250
	3	188.324	194.221	195.137	172.594	186.378
3	2	206.452	184.475	220.807	214.949	211.114
	3	212.549	205.075	210.255	200.292	222.580

6.2. Parameter Sensitivity Analysis

Within this section, we examine the model's parameter sensitivity and investigate the influence of the graph neural network's layer count, attention head count, and encoder layer count on the quality of the solution. The training loss for various parameters is depicted in Figure 5. We adjusted the number of GNN layers to 2, 4, and 6, respectively, in order to retrain the model. It is evident from Figure 5 that when the number of layers is increased, the convergence effect is not substantial, indicating that increasing the number of GNN layers has no significant positive effect on the model's convergence. The sensitivity of the number of layers is negligible. We adjusted the quantity of attention heads to 2, 4, and 6 and carried out training in that order. The model's convergence performance slightly deteriorated with the increase in the number of attention heads, but when it continued to increase, there was no obvious difference in the effect. It is noteworthy that an increase in the number of attention heads will lead to an increase in the number of parameters and calculations. By adjusting the encoder layers to 1, 3, and 5, we observed a significant impact on the training outcome when the number of encoder layers is 1. The shallower network structure is not conducive to network training, which affects the final solution quality of the model. When the number of encoder layers increases and the encoder structure has 3 layers, the training of the model converges faster, and the quality of the solution also increases. Nevertheless, the impact is only marginally enhanced when the number of encoder layers

is consistently augmented to 5 layers. The selection of deep learning model layers requires a judicious trade-off between training performance and computational complexity.

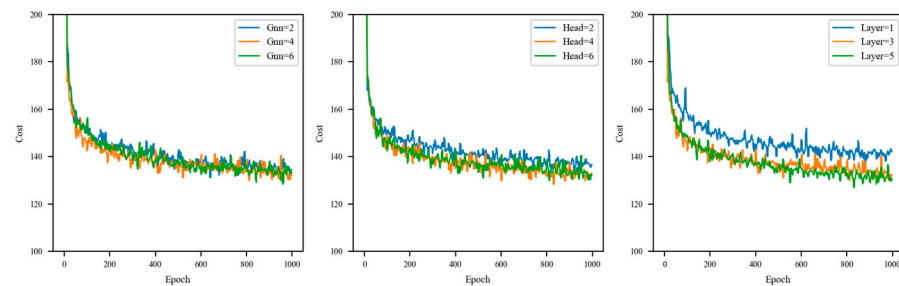


Figure 5. Training Loss Curves for Sensitivity Analysis of GNN Layers, Attention Heads, and Encoder Layers.

7. Discussion

The DRL model proposed in this study exhibits a “scale-dependent performance inversion” characteristic, strongly supporting the research hypothesis that “DRL can balance solution quality and efficiency in MDVRPSTW.” In small-scale problems, although its objective value is slightly higher than the other two algorithms, the gap with GVNS (averaging 8.9%) remains small, while the solution time is reduced by 96.8%, maintaining practical application competitiveness. In medium and large-scale problems, the DRL model achieves comprehensive superiority: at $N = 50$, the objective value is 27.7% lower than GVNS and 47.8% lower than VRPSolverEasy; at $N = 100$, the objective value is 96.3% lower than GVNS and 75.4% lower than VRPSolverEasy (excluding timeout cases). This advantage stems from the model’s encoder–decoder architecture—graph neural networks capture topological associations between depots and customers, while multi-head attention fuses spatiotemporal features [18], enabling simultaneous optimization of multiple vehicle paths and avoiding the constraints of traditional “divide-and-conquer” strategies. Additionally, the DRL model maintains stable solution times across scales ($N = 20$: average 3.51 s; $N = 50$: average 6.09 s; $N = 100$: average 4.29 s), thanks to the “offline training–online decision-making” paradigm of DRL [17]—strategy networks trained with large-scale random data can directly output routing paths for new problem cases without re-exploring the solution space, perfectly matching the real-time demands of fresh e-commerce forward warehouse distribution.

These findings transcend mere algorithm performance comparisons. Academically, this study fills the research gap in DRL-based solutions for the MDVRPSTW problem, demonstrating that the Markov Decision Process (MDP) model integrating global static states with vehicle dynamics effectively handles complex constraints like multi-vehicle depots, multiple vehicles, and soft time windows. Industrially, the stable and efficient performance of the DRL model provides a viable technical solution for optimizing forward warehouse distribution in fresh food e-commerce enterprises. Compared to traditional algorithms, it reduces total distribution costs by 27.7% to 96.3% and shortens response times from hours/minutes to seconds, addressing the pain points of high cold chain costs and stringent delivery timeliness constraints for fresh products.

8. Conclusions

This section summarizes the key findings of the experimental results. The deep reinforcement learning model proposed in this paper demonstrates outstanding performance in solving the MDVRPSTW problem under the front-end warehouse model. For small-scale problems ($N = 20$), the model maintains strong competitiveness, and its solution time is

reduced by 96.8% and 96.2% compared with GVNS and VRPSolverEasy, respectively, and there is only a 8.9% difference from the GVNS target value.

In medium-scale and large-scale problems ($N = 50/100$), the model demonstrates comprehensive superiority in both solution quality and efficiency, achieving 27.7% to 96.3% lower objective values compared with the baseline algorithms, with solution times consistently maintained between 3 and 10 s, effectively avoiding the exponential growth characteristic of traditional algorithms.

The model demonstrates exceptional adaptability to complex configurations: as the number of depots and vehicles increases, the target value decreases continuously (with a maximum reduction of 28.99%), while the solution time shows no exponential growth, fully meeting the diverse configuration requirements of forward warehouse distribution.

In conclusion, the DRL model effectively balances solution quality and efficiency for the MDVRPSTW problem, providing a novel technical approach for optimizing distribution in fresh e-commerce forward warehouses. It fills the gap in applying deep reinforcement learning to MDVRPSTW and lays the groundwork for future research on more complex vehicle routing problems.

While this study simplifies vehicle time variations, resulting in discrepancies from real-world scenarios, it remains unsuitable for time-sensitive applications. Future research could focus on three key areas: (1) incorporating dynamic demand constraints (e.g., real-time order fluctuations) to enhance model adaptability to actual delivery scenarios; (2) optimizing the distance-time conversion ratio between nodes to 1:1; and (3) developing lightweight model architectures to reduce training resource consumption and facilitate deployment on edge devices such as logistics delivery vehicles.

Author Contributions: M.J.: Conceptualization, methods, writing. G.C.: content. J.C.: validation. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data Availability Statement: Additional data available in Table 4.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Liu, X.; Gou, X.; Xu, Z. Multi-Objective Last-Mile Vehicle Routing Problem for Fresh Food E-Commerce: A Sustainable Perspective. *Int. J. Inf. Technol. Decis. Mak.* **2024**, *23*, 2335–2363. [\[CrossRef\]](#)
2. Huang, Y. Evolutionary game analysis of collaborative strategies in fresh e-commerce and cold chain logistics: The role of incentive mechanisms and supervision policies. *Int. Rev. Econ. Financ.* **2025**, *104*, 104773. [\[CrossRef\]](#)
3. Li, X.; Luo, H.; Wang, G.; Song, Z.; Gou, Q.; Meng, F. Optimizing multi-drone patrol path planning under uncertain flight duration: A robust model and adaptive large neighborhood search with simulated annealing. *Appl. Soft Comput.* **2025**, *176*, 113107. [\[CrossRef\]](#)
4. Guo, E.; Gao, Y.; Hu, C. A two-stage evolutionary algorithm based on hybrid penalty strategy and its application to multi-UAV path planning. *Expert Syst. Appl.* **2025**, *298*, 129698. [\[CrossRef\]](#)
5. Hung, P.D.; Tam, N.T.; Binh, H.T.T. Adaptive ant colony optimization for solving dynamic vehicle and drone routing with time window constraints. *Evol. Intell.* **2025**, *18*, 110. [\[CrossRef\]](#)
6. Wang, S.; Sun, W.; Huang, M. An adaptive large neighborhood search for the multi-depot dynamic vehicle routing problem with time windows. *Comput. Ind. Eng.* **2024**, *191*, 110122. [\[CrossRef\]](#)
7. Wang, Y.; Chen, P. An adaptive large neighbourhood search for multi-depot electric vehicle routing problem with time windows. *Eur. J. Ind. Eng.* **2024**, *18*, 606–636. [\[CrossRef\]](#)
8. Bezerra, S.N.; Souza, M.J.F.; de Souza, S.R. A variable neighborhood search-based algorithm with adaptive local search for the Vehicle Routing Problem with Time Windows and multi-depots aiming for vehicle fleet reduction. *Comput. Oper. Res.* **2022**, *149*, 106016. [\[CrossRef\]](#)
9. Rabbani, M.; Pourreza, P.; Farrokhi-Asl, H.; Nouri, N. A hybrid genetic algorithm for multi-depot vehicle routing problem with considering time window repair and pick-up. *J. Model. Manag.* **2018**, *13*, 698–717. [\[CrossRef\]](#)

10. Anuar, W.K.; Lee, L.S.; Seow, H.-V.; Pickl, S. A Multi-Depot Dynamic Vehicle Routing Problem with Stochastic Road Capacity: An MDP Model and Dynamic Policy for Post-Decision State Rollout Algorithm in Reinforcement Learning. *Mathematics* **2022**, *10*, 2699. [[CrossRef](#)]
11. Arishi, A.; Krishnan, K. A multi-agent deep reinforcement learning approach for solving the multi-depot vehicle routing problem. *J. Manag. Anal.* **2023**, *10*, 493–515. [[CrossRef](#)]
12. Zhang, K.; Lin, X.; Li, M. Graph attention reinforcement learning with flexible matching policies for multi-depot vehicle routing problems. *Phys. A Stat. Mech. Appl.* **2023**, *611*, 128451. [[CrossRef](#)]
13. Errami, N.; Queiroga, E.; Sadykov, R.; Uchoa, E. VRPSolverEasy: A Python Library for the Exact Solution of a Rich Vehicle Routing Problem. *INFORMS J. Comput.* **2024**, *36*, 956. [[CrossRef](#)]
14. Masmoudi, M.; Borchani, R.; Jarboui, B. Generalized variable neighborhood search algorithm for vehicle routing problem with time windows and synchronization. *Comput. Oper. Res.* **2025**, *183*, 107193. [[CrossRef](#)]
15. Rios, B.H.O.; Xavier, E.C. Metaheuristic approaches for the stochastic capacitated multi-depot vehicle routing problem with pickup and delivery. *Expert Syst. Appl.* **2025**, *290*, 128258. [[CrossRef](#)]
16. Chen, S.; Yin, Y.; Sang, H.; Deng, W. A hybrid GRASP and VND heuristic for vehicle routing problem with dynamic requests. *Egypt. Inform. J.* **2025**, *29*, 100638. [[CrossRef](#)]
17. Guan, Q.; Xue, S.; Tan, J.; Jia, L.; Cao, H.; Chen, B. Dynamic embedding-based deep reinforcement learning for heterogeneous capacitated VRPs with unloading time constraints. *Expert Syst. Appl.* **2025**, *293*, 128660. [[CrossRef](#)]
18. Cai, H.; Xu, P.; Tang, X.; Lin, G. Solving the Vehicle Routing Problem with Stochastic Travel Cost Using Deep Reinforcement Learning. *Electronics* **2024**, *13*, 3242. [[CrossRef](#)]
19. Wang, Y.; Hong, X.; Wang, Y.; Zhao, J.; Sun, G.; Qin, B. Token-based deep reinforcement learning for Heterogeneous VRP with Service Time Constraints. *Knowl.-Based Syst.* **2024**, *300*, 112173. [[CrossRef](#)]
20. Solomon, M.M. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Oper. Res.* **1987**, *35*, 254–265. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.