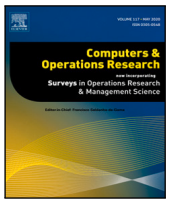
Computers & Operations Research 184 (2025) 107217

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor



Knowledge-guided hybrid deep reinforcement learning for the dynamic multi-depot electric vehicle routing problem

Reza Shahbazian, Alessia Ciacco, Giusy Macrina, Francesca Guerriero *

Department of Mechanical, Energy and Management Engineering - DIMEG, University of Calabria, Rende, 87036, Italy

ARTICLE INFO

Dataset link: Github Repository

Keywords:
Knowledge-guided learning
Reinforcement learning
Deep learning
Vehicle routing problem
Variable neighborhood search

ABSTRACT

In this paper, we consider the dynamic multi-depot electric vehicle routing problem with time windows, proposing a novel hybrid framework, integrating knowledge-guided multi-agent deep reinforcement learning (MARL) and a variable neighborhood search (VNS) algorithm. The MARL component employs a double-deep Q-network for initial route generation, which is further refined by the VNS to enhance solution quality. Real-time decision-making and adaptive optimization enable the framework to respond effectively to dynamic changes in the environment, leading to improved efficiency, reduced costs, and enhanced overall performance. Extensive experiments on both synthetic and real-world benchmark datasets, demonstrate the framework's superiority over state-of-the-art algorithms, showing significant improvements in total traveled distance, computation time, and scalability. The results indicate over 70% reduction in the average total traveled distance compared to state-of-the-art baselines on small-scale datasets. Importantly, the framework's ability to handle large-scale problems effectively makes it a promising solution for real-world applications.

1. Introduction

The growing need for cost-effective, safe and environmentally friendly transportation has led to growing attention to the development of intelligent autonomous vehicles (Kopelias et al., 2020; Ajao and Oludamilare, 2023). Given technical advances, autonomous electric vehicles are expected to change the way in which customer demand is satisfied (Kucukoglu et al., 2021). This new paradigm requires the development of effective route planning approaches, in order to minimize transportation and delivery costs, leading to the modeling and solving of vehicle routing problems (VRPs) (Zhu et al., 2023).

The original version of the VRP, presented by Dantzig and Ramser (1959), is an optimization problem, with a set of scattered customers associated with demand and a number of vehicles departing from a central depot. The primary goal of the VRP is to identify the routes with the lowest possible cost, which usually includes the total distance or time of the routes, by which all customers are visited and severed merely once. Variants of this problem have emerged over time to present and accommodate different operational constraints (Sar and Ghadimi, 2023; Kucukoglu et al., 2021). For example, VRP with time windows constraint (VRPTW) aims to serve customers within their specific time windows (Zong et al., 2024). The recently introduced electric vehicle routing problem (EVRP) focuses on creating efficient route plans for vehicles, while satisfying various battery-related constraints (Macrina et al., 2019b,a; Kucukoglu et al., 2021). Despite the number of existing studies in EVRP, there is still a need for methods that handle the routing problem effectively and efficiently, particularly in dynamic scenarios and when time windows constraints need to be satisfied (Raja et al., 2023). The VRPs can be classified into four primary categories based on the availability of information and the inherent uncertainty, namely deterministic, dynamic, stochastic, and dynamic-and-stochastic VRPs (Ritzinger et al., 2016). In deterministic VRPs, all relevant information is available in advance, while dynamic VRPs incorporate information that evolves over time. Consequently, solutions for dynamic VRPs must be modified in real-time as new information emerges. On the other hand, stochastic VRPs address uncertainty in parameters such as travel times or demands, and dynamic-and-stochastic routing problems integrate both dynamic and stochastic components, thereby representing the most intricate scenario. VRP and its variants have received great attention, classified as NP-hard problems in combinatorial optimization (Lenstra and Kan, 1981; Kim et al., 2015). Most of the existing literature on VRPs focuses on modeling deterministic scenarios, while stochastic and dynamic configurations have received less attention (Sar and Ghadimi, 2023).

* Corresponding author.
E-mail addresses: reza.shahbazian@unical.it (R. Shahbazian), alessia.ciacco@unical.it (A. Ciacco), giusy.macrina@unical.it (G. Macrina), francesca.guerriero@unical.it (F. Guerriero).

https://doi.org/10.1016/j.cor.2025.107217
Received 3 January 2025; Received in revised form 13 April 2025; Accepted 17 July 2025
Available online 28 July 2025

The existing solution approaches proposed for VRPs primarily rely on approximation and heuristic methods. Among these, local search-based algorithms, have garnered significant attention in recent years (Bezerra et al., 2023). Local search algorithms repeatedly apply different search operators to develop and modify an initially created solution, aiming to achieve one of higher quality. It is evident that the quality of the final solution produced by local search algorithms largely depends on the initial solution. If the initial solution is poorly suited to the problem, it can lead to suboptimal outcomes (Zirour, 2008).

On the other hand, reinforcement learning (RL)-based algorithms have shown promising results in combinatorial optimization problems in recent years (Bello et al., 2016; Nazari et al., 2018; Hildebrandt et al., 2023). These RL-based methods are effective, scalable to large-scale problems, and are able to quickly infer and handle route generation (James et al., 2019; Li et al., 2018). However, the majority of the RL-based algorithms proposed for VRPs suffer from a lack of robust, high-quality solutions in comparison to the heuristics (Shahbazian et al., 2024; Bogyrbayeva et al., 2024). In addition, there are some inherent challenges regarding RL-based solutions, including the growth of state–action space due to the rise of problem complexity, the potential time-consuming training process, and numerous interactions with the environment.

In recent years, domain knowledge has been integrated into machine learning (ML) models. In particular, scientific knowledge-guided machine learning (KGML) is a new area of ML research that uses scientific theories to direct the development and training of ML models for better generalizability and scientific consistency (Willard et al., 2022; Karpatne et al., 2017). The RL models can also benefit from integrating the extra knowledge into the learning process by guiding the exploration–exploitation of the agents. Particularly in model-free RL techniques, where agents have limited understanding of the environment, knowledge-guided RL can assist agents in exploration. This guidance accelerates the learning process, leads to more appropriate policies, enhances the model performance in real-world settings, and helps avoid agents moving towards undesirable states. This additional knowledge can be integrated into various stages of the RL pipeline, such as *problem representation*, *learning strategy*, *task structuring*, and *simulation-to-real-world deployment* (Eßer et al., 2022).

In this work, we explore the dynamic multi-depot electric vehicle routing problem with time windows (DMDEVRPTW), proposing a hybrid framework that brings together both deep reinforcement learning (DRL) and local search. Moreover, by integrating the external domain-knowledge into our DRL component, our framework can provide improved solution quality, faster convergence, and better generalization. The dynamic nature of the VRPs is modeled as the unpredictability of real-time factors that may change unexpectedly or become known only when vehicles reach the customers. As a result, the model needs to evaluate the current situation and customer demands in real-time to make timely decisions consistently (Ritzinger et al., 2016; Erera et al., 2010; Chang, 2005). Our proposed framework is cost-efficient and provides satisfactory solutions for large-scale problems. In our proposed model, we use the DRL outcome as the initial solution for the local search algorithm to further enhance the final route planning and reduce the computation time. Unlike previous studies, we consider both multi-depot scenarios and the dynamism in demands and environments. In particular, we assume that the demands of the customers, travel time, energy usage and service time may all be subject to unexpected changes during the route execution. Furthermore, we investigate the potential of a property called *state–action permissibility (SAP)* (Mazumder et al., 2018, 2022) by integrating SAP-inspired knowledge into our RL-based proposed framework. This integration not only accelerates the training process, but also guides the model towards a better solution. We use a geographic partitioning approach to further define the execution of our SAP mechanism. To do this, the environment is pre-partitioned before the routing process starts, and each depot is assigned to its particular geographical area. Throughout the process, the current location of each customer and vehicle is dynamically evaluated based on the distance to its assigned depot. Then, the SAP mechanism can screen possible customer visits, enabling the vehicle only to advance to consumers found in its given area or take actions in proximity to its assigned depot. This procedure reinforces a domain-specific heuristic, therefore reflecting the ideals of knowledge-guided machine learning by guaranteeing that vehicles give actions first that match with effective depot-centric routing.

To illustrate, as shown in Fig. 1, we consider a fleet of vehicles to transport items from various locations (multi-depot) to a set of customers located in a given area. The main aim is to identify the optimal route for each vehicle, while minimizing the overall distance traveled and satisfying time limits for each delivery.

In the considered scenario, we require a solution that accounts for both pre-known constraints and system dynamism. Additionally, the solution must be optimal and computationally efficient for large-scale problems.

This paper's contributions are summarized as follows:

- We propose a knowledge-guided DRL-based approach by employing a double-deep Q-Network (DDQ) architecture. We use a neural network to estimate the optimal Q-function, and a prioritized experience replay (PER), which can simultaneously speed up convergence while enhancing the performance of the model.
- To further enhance the solution quality, we develop a hybrid DRL and local neighborhood search (LNS) framework that benefits from both RL and heuristics to provide high-quality solutions in reasonable time for large-scale routing problems.
- We perform comprehensive experiments on real-world data and analyze several evaluation scenarios. We consider both mid-sized and large-scale VRP instances, and compare our proposed approaches to state-of-the-art solutions.

The remainder of the paper is organized as follows. Section 2 provides an overview of related works. Section 3 describes the studied problem and its formulation as a Markov Decision Process (MDP). The key components of the proposed solution strategy are detailed in Section 4, while the developed framework is presented in Section 5. Section 6 discusses the experimental study. Finally, the paper concludes with final remarks in Section 7.

2. Related works

The solution approaches for VRPs can be categorized into exact methods, heuristics and machine-learning-based methods. The exact approaches provide an optimal solution, such as those introduced by Sluijk et al. (2023), Caceres-Cruz et al. (2014). However, despite their effectiveness in small-scale problems with few constraints, such methods are unable to provide the expected solution in acceptable time complexity when the number of constraints or the scale of the problem increases. For instance, IBM CPLEX solver cannot provide a feasible solution for constraint VRPs with more than 30 customers in an acceptable time (Macrina et al., 2017). Heuristics and meta-heuristic strategies accelerate exact methods in reaching a solution. Compared to the exact methods, heuristics perform well, especially when dealing with mid-sized problems. Machine learning-based solutions fall into the third category. These can be used on their own to solve VRPs or with heuristics to fine-tune their parameters (Laporte and Semet, 2002; Ammar et al., 2022; Liu et al., 2023; Bouanane et al., 2022). Reinforcement learning is the most promising solution in this category (Bai et al., 2023).

In what follows, we cover merely the most related studies focusing on RL-based or LNS algorithms to solve multi-depot VRPTW, or those that model the uncertainties in the EVRP. An overview of the covered related works is presented in Table 1.

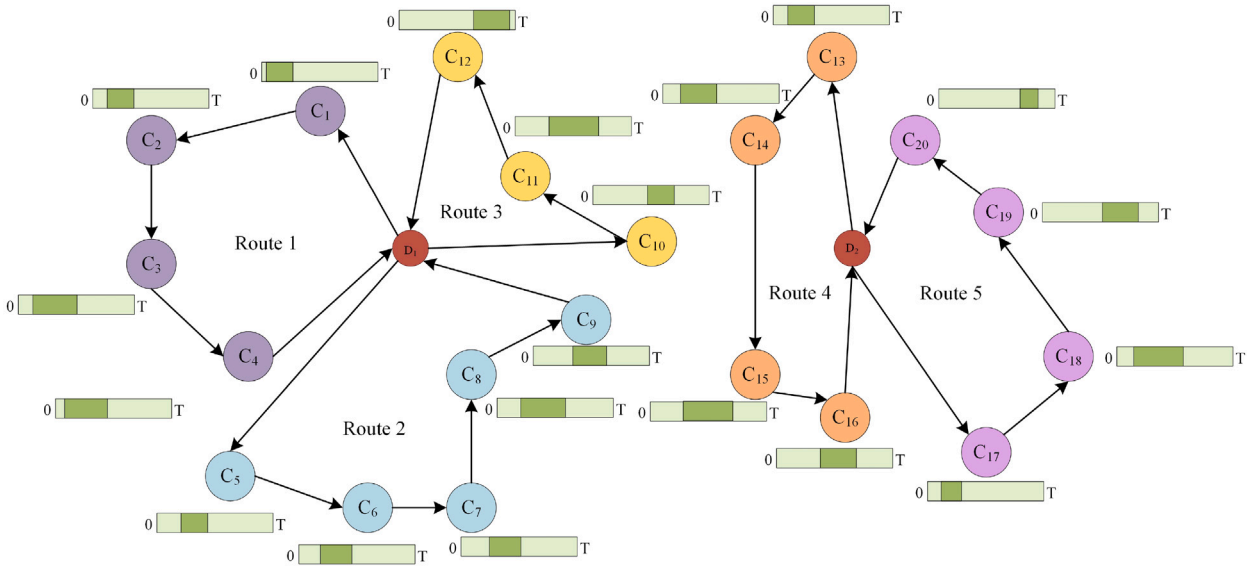


Fig. 1. An instance of DMDEVRPTW problem with 20 customers, 2 depots and their routes.

Table 1
Overview of the related works in the literature.

Reference	Formulation	VRP Variant	Approach	Fleet	Objective	Depot	Time Window	Instances	Max Problem Size
(Nazari et al., 2018) (2018)	MDP	CVPR	RL-Att	Single	TD	Single	No	Rdm	100
(Basso et al., 2021) (2021)	MILP	EVRP-CC	PBML	Homogeneous	TD, TEC	Single	No	Rdm, RW, Sim	20
(Qin et al., 2021) (2021)	MILP	RLHH		Heterogeneous	TEC	Single	No	Rdm, Benchmark	100
(Li et al., 2022) (2022)	MDP	HCVRP	DRL	Heterogeneous	TT	Single	No	Rdm, CVRPLib	160
(Alcaraz et al., 2022) (2022)	DP	VRPTW	DRL+MCTS		DG+DR+DT	Single	Y	RW, Sim	-
(Xu et al., 2022) (2022)	Graph (*MDP)	(PC)TSP, CVRP, SDVRP, OP	Att-RL	Single	TD, TP	Single	No	Rdm	100
(Lin et al., 2022) (2022)	Graph	EVRPTW	DRL	Homogeneous	TD	Single	Y	Rdm,	100
(Phiboonbanakit et al., 2021) (2021)	Graph (*MDP)	SDVRP	RL+TR		TD	Single	Y	Nazari et al. (2018), CS	50
(Pan and Liu, 2023) (2023)	POMDP	DU-VRP	Att-DRL	Heterogeneous	TT	Double	No	Rdm, RW(OSM)	100,200
(Tan et al., 2023) (2023)	Uncertainty Model	RMOVRPTW	MOD, GO, MCTs		TD,NV	Double	Y	Solomon	100
(Zong et al., 2024) (2024)	Graph (*MDP)	VRPTW	RL+ED(MHA)	*Heterogeneous	TD	Single	Y	RW	1000
(Chen et al., 2023) (2023)	MDP	VRPMDP	DRL	Single	Distance	Single	No	Rdm, RW	100
(Bogyrbayeva et al., 2023) (2023)	MDP	TSP-D	DRL-ED-Att	Heterogeneous	TT	Single	No	Random, (Haider et al., 2019)	100
(Ren et al., 2022) (2022)	MDP	MVRPSTW	MARL+-ED-Att	Homogeneous	TD, TP	Single	Y	Rdm,	100
(Zhang et al., 2020b) (2020)	Graph (*MDP)	MVRPSTW	DRL, ED-Att	Homogeneous	TD, TP	Single	Y	Rdm	150
(Wang et al., 2024) (2024)	Graph	MD-DVRPTW	ALNS	Homogeneous	Distance	Multi	Y	*Cordeau*	288
(Zhao et al., 2020) (2020)	Graph (*MDP)	VRPTW	DRL(AC)+LNS	Homogeneous	TD	Single	Y	Rdm	100
(Stodola and Nohel, 2022) (2022)	Graph (*MDP)	MVRP	ACO	*(Homogeneous)	TD	Multi	No	Cordeau	360
(da Silva Junior et al., 2021) (2022)	Graph (*MDP)	DVRPTW	ACO	Homogeneous	TD/NUsC	Single	Y	Extention of Solomon	100
(Zhang et al., 2023) (2023)	MDP	DVRPSR	Knps-L	*Homogeneous	NSR	Single	Request arrival time	Modified OSM	-
(Dubey and Tanksale, 2023) (2023)	Multi-graph	MDVRP-TW-SP-SD	GA+LS	Heterogeneous	VHC, VRC, PC,	Multi	Y	Modified Cordeau, CS	288
(Wang et al., 2023b) (2023)	bi-objective MILP	MDVRPTW-TDLC	GA	Homo/Heterogeneous	VHC, VVRC, VLR	Multi	Y	Modified Cordeau	288
(Quirion-Blais and Chen, 2021) (2021)	Multi-objective MIP	VRPTW	CBR	*Homogeneous	LCCh, VHC, VRC, TP	Single	Y	Rdm, RW CS	134
(Baty et al., 2024) (2024)	Graph	VRPTW	ML-OC	Homogeneous	TP,TRC	Single	Y	RW	100
This study	MDP	DMDEVRPTW	KDRL+VNS	Homogeneous	TD,TP	Multiple	Y	Rdm, RW	960

A star (*) indicates that these aspects are not explicitly mentioned in the respective papers.
Model: *POMDP:partially observed Markov decision process; MIP: Mixed-Integer Programming; MILP:Mixed-Integer Linear Programming; DP: Dynamic Programming;OP: Orienting Problem; RMOVRPTW: Robust Multi-objective VRPTW*
Objective: *TD: Total Distance; TP: Time Penalty; NV: number of vehicle; NUsC: Number of unserved customers; NSR: Number of Served Requests; TT: Total Time; TEC: Total Energy Consumption; VHC: vehicle hiring cost; VRC: vehicle routing cost; PC: penalty cost; VLR: vehicle loading rate; TP: Total Profit; TRC: Total Routing Cost;DG: Delivered Goods; DR: Delivery Rest Reductions; DT: Driving Time Extentions; LCCh: Length of customer chains;*
Approach: *Att:Attention; TR:Tree-based regression; ED: Encoder–Decoder; MHA: Multi-head attention; ACO: Ant Colony; Knps-L: knapsack-based linear models; ML-OC: Machine Learning based Combinatorial Optimization; CBR: Case base Reasoning; PBML:probabilistic Bayesian machine learning; RLHH:RL-based Hyper-heuristic; MCTS: Monte Carlo tree search; GO: Genetic Operators; MOD:Multi-objective decomposition; MCTs: Monte-Carlo tests*
Dataset: *Rdm:Random; RW:real world; OSM:OpenStreetMap; Sim:Simulation; CS:Case Study.*

2.1. RL-based approaches

Nazari et al. (2018), proposed one of the first promising RL-based solutions for VRPs. They assumed that the observation is limited to the reward signals and proposed a parameterized stochastic policy and gradient algorithm to optimize the parameters of that policy. Their evaluation results confirm their method's efficiency in mid-sized instances compared to classical heuristics and Google-OR tools. However, they consider single-depot VRP with known parameters. In Kalakanti et al. (2019), the authors discuss the scalability issues and realistic simulation settings of customer environments. They then propose an RL-based algorithm that uses a two-phase solver and geometric clustering.

Basso et al. (2021) study EVRP by considering uncertainty in the traffic. They propose a two-stage probabilistic Bayesian ML approach by first finding the best paths and then optimizing the routes, considering the uncertainty in energy demand by planning to charge within a confidence interval. Their experimental results on real data from Gothenburg-Sweden for 24-hour traffic in the city of Luxembourg indicate high accuracy for energy prediction as well as energy savings and more reliability for the routes compared to non-ML-based methods.

Qin et al. (2021) study the heterogeneous VRP in which the vehicles have different capacities, mathematically formulating the problem as mixed-integer linear programming and providing the optimal solution for small-scale instances. To be able to handle large-scale instances, they propose a RL-based hyper-heuristic in which meta-heuristics are used for policy selection. The authors in Li et al. (2022) study the heterogeneous VRP in which the vehicles have different capacities. They consider min–max and min-sum objectives to minimize the longest or total travel time and propose a deep RL algorithm. Their evaluations are performed on randomly generated instances of VRP and are compared to heuristics, which shows the acceptable performance of their proposed method. Alcaraz et al. (2022) investigate the uncertain travel and service times in VRP. They assume that incidents might introduce additional delays and that the predefined schedules become inadequate, and they consider the driver's behavior in terms of driving time, breaks, and rest periods, proposing an RL-based strategy that combines roll-out and Monte Carlo tree search for decision-making on

a trained offline model. Xu et al. (2022) study the application of RL to the VRP and propose a new attention-based RL model using batch normalization, reordering, and gate aggregation. Lin et al. (2022) study EVRP with time windows and propose an end-to-end deep RL method, parameterizing a stochastic policy, and training the model using a policy gradient.

Phiboonbanakit et al. (2021) propose an optimization model based on RL and a tree-based regression method for route optimization and feasibility evaluation, respectively. Pan and Liu (2023) study real-time tracking for urban logistics and propose a DRL framework by considering the uncertainty in the VRP. They model the problem as an MDP, use a deep neural network with a dynamic attention mechanism, and evaluate their proposed algorithm on randomly generated instances, in which the customers are randomly located on an 1×1 area and one depot, assuming that the customers' demand is a random variable parameter and that the vehicles have different capacities. Tan et al. (2023) study VRPTW with uncertainty, proposing a robust algorithm to optimize the total distance and the number of vehicles required to make the deliveries. They prioritize the customers to form feasible routes, use Monte-Carlo tests to evaluate the found solutions and use Solomon's benchmark for evaluations.

Zong et al. (2024) pointed out the highlighted challenges in VRPs, and proposed an end-to-end RL-based framework, that contains encoding the problem's constraints into features, a time penalty augmented reward, and a task handler for ensuring vehicle cooperation.

In their study, Chen et al. (2023) studied the VRP with mixed delivery and pickup (VRPMDP). They came up with an RL graph neural network-based encoder–decoder framework with a coordinated decision of loading and routing mechanism.

Bogyrbayeva et al. (2023) considered using attention-based models in traveling salesman problem (TSP) with drones (TSP-D). They considered the coordination-oriented issue when a heterogeneous fleet of vehicles is in operation. The authors developed a mixed model that uses an attention encoder and a long short-term memory (LSTM) network decoder. In Lei et al. (2022), the authors propose an end-to-end DRL framework to effectively solve TSP and VRPs with limited capacity (CVRP). Turning to multi-agent RL-based methods, the authors in Ren et al. (2022) proposed a model for the VRPTW with soft time windows constraints (VRPSTW) in modern supply chains, using a self-attention mechanism-based encoder–decoder framework to generate routes for each vehicle iteratively, and multiple route recorders to extract route history information and enable communication between vehicles. A multi-sampling strategy is also used to balance computation time and performance improvement. The reinforcement learning method is used to train the neural network, with a self-critic baseline to reduce variance. Zhang et al. (2020b) proposed a multi-agent attention model that uses RL to effectively resolve VRPSTW in urban logistics distribution systems. Their model uses an encoder–decoder framework with attention layers to generate tours of multiple vehicles iteratively. The results show that the proposed method could outperform traditional methods and Google OR-Tools with minimal computation time. Additionally, varying customer numbers and vehicle capacities validate the model's robustness.

2.2. LNS-based approaches

The study carried out by Wang et al. (2024) examines the multi-depot dynamic vehicle routing problem with time windows (MDDVRPTW), where customer requests emerge stochastically throughout the operational horizon. They develop an adaptive large neighborhood search (ALNS) re-optimization framework to provide timely and comprehensive service to these customers, using two removal operators and a time windows compatibility-based insertion method. Their study demonstrates that the ALNS can be well-suited for dynamic problems and re-optimization tasks within a short period of time.

He et al. (2021) introduced an adaptive variable neighborhood search ant colony algorithm (AVNSACA) to solve the VRPSTW, improving the ant colony algorithm's pheromone update strategy, and designing two variable neighborhood search operators to prevent falling into local optimum. The Solomon benchmark problem is used to verify the effectiveness of the proposed algorithm, and the comparative analysis of the two algorithms demonstrates the advantages of the improved ant colony algorithm. Hottung and Tierney in Hottung and Tierney (2022) studied the CVRP and the split delivery vehicle routing problem (SDVRP), and proposed a LNS framework that integrates learned heuristics for generating new solutions. The learning mechanism is based on a deep neural network with an attention mechanism, designed to be integrated into an LNS search setting, resulting in outperforming the existing machine learning approaches and coming close to the state-of-the-art optimization approaches. Zhao et al. (2020) develop a hybrid model of DRL which comprises an actor, an adaptive critic, and a routing simulator, to generate routing strategies and change network structure to accelerate convergence. The DRL model is then combined with a local search method to improve solution quality of the obtained routes. Their evaluations of LNS algorithms suggest potential applications in real-world transportation. However, these algorithms may be limited in their ability to solve stochastic and dynamic problems. Therefore, extending heuristics by combining them with other methods could be a promising approach to enhance the performance of such algorithms. Attempting to address the issues of the existing multi-stage stochastic and dynamic optimization approaches, such as being computationally expensive or struggling on high-dimensional combinatorial problems, Baty et al. (2024) developed an ML-based pipeline that incorporates a combinatorial optimization layer to solve the dynamic VRPTW. A comprehensive numerical study shows that the policy encoded via the CO-enriched ML pipeline outperforms greedy policies by 13.18% and even Monte Carlo policies by 1.57% in terms of travel time on average.

Existing research places limited emphasis on multi-depot scenarios, with only a few studies directly addressing MDVRPs. Moreover, the customer's needs, time windows, and travel times are mainly treated in a deterministic or resilient manner. Therefore, considering these uncertainties and dynamism in a RL-based framework can improve robustness and practicability. Furthermore, due to the potential of unexpected changes, such as unforeseen weather conditions or customer changes in demand or cancellations, this approach can be more effective and applicable in real-world scenarios. Therefore, we address both of these identified shortcomings in this study.

3. Problem statement

The DMDEVRPTW involves planning efficient delivery routes for a homogeneous fleet of electric vehicles (EVs) operating from multiple depots, requiring real-time decision-making in dynamic, time-constrained environments. Each EV in the fleet has a limited battery capacity and can be recharged at designated charging stations, ensuring that the vehicle's battery remains positive throughout the routes (Keskin et al., 2021). In contrast to deterministic routing problems, the DMDEVRPTW framework operates in a dynamic environment subject to possible changes. In this context, new customer requests may emerge, existing requests may be modified or cancelled, and unforeseen events, such as traffic congestion or vehicle malfunctions, can disrupt pre-established routes. In the scenario under consideration, customer demand, travel time, energy consumption, and service time are assumed to be revealed only during route execution, specifically when the EV arrives at the customer's location (Iklassov et al., 2023; Ma et al., 2023). This significantly increases the problem's complexity, as it affects the feasibility of planned routes and the overall efficiency of the delivery operation. Despite its complexity, this problem arises in a variety of real-world applications, including logistics, transportation, and delivery services. The primary objective is to minimize the total

cost of serving the customers. The problem also involves balancing competing objectives, such as minimizing travel distance, ensuring that all customer demands are satisfied within their time windows and that the battery capacity of each EV is sufficient to complete its route (Lin et al., 2021; Zhang et al., 2020a). Therefore, the DMDEVRPTW problem presents numerous challenges that must be addressed to develop effective solution approaches. Real-time algorithms must be capable of quickly adapting to changes, generating new routes, and reoptimizing existing ones. Additionally, the computational complexity of the problem, especially for large-scale instances, requires efficient solution techniques.

A major challenge arises from the dynamic nature of the problem, where parameters such as customer demand, travel time, service time, and energy consumption are unknown beforehand. This complicates the planning of optimal routes, as actual values may deviate from initial estimates, and unexpected variations can disrupt the planned schedule (Iklassov et al., 2023). Another significant challenge is the limited battery capacity of EVs, which requires careful route planning to ensure that vehicles have sufficient energy to complete their assigned routes. Furthermore, the availability of charging stations adds another layer of complexity to the problem.

Traditional approaches often struggle to efficiently address the complexities of the DMDEVRPTW problem. These methods typically rely on deterministic assumptions about problem parameters, while in real-world scenarios, these parameters are often subject to uncertainty and unexpected changes, making it difficult for traditional methods to generate robust and efficient solutions. To address the challenges posed by the DMDEVRPTW, we propose a novel hybrid framework that combines the power of Knowledge-Guided Multi-Agent Deep Reinforcement Learning (MARL) and Variable Neighborhood Search (VNS). This integrated approach offers a robust and efficient solution to the problem, successfully dealing with challenges associated with the DMDEVRPTW problem. RL is an ideal approach for DVRPs due to its ability to learn optimal decision-making strategies in dynamic and uncertain environments. RL agents can adapt to real-time changes, such as unexpected traffic or customer requests, and continuously improve their performance through experience. This makes RL a powerful tool for solving complex DVRPs, especially in large-scale and real-world scenarios.

Leveraging the MARL, the proposed framework provides robustness to uncertainties in the environment, by learning from interactions and adapting to deviations in the environment. By training on a diverse set of scenarios, the designed DDQN can learn to make robust decisions, such as selecting optimal routes and charging strategies, even in the face of uncertainty (Wang et al., 2023a). This component is further enforced by aggregating external knowledge of the environment, better guiding the model towards high quality solutions. Integrating external knowledge into MARL significantly enhances its performance in solving VRPs. By incorporating domain-specific information, agents can make more informed decisions and learn more efficiently. This leads to improved solution quality, faster convergence, and better generalization, ultimately resulting in more robust and intelligent solutions for complex VRP problems. The MARL component also effectively handles multiple depots by learning to assign vehicles to appropriate depots and plan efficient routes from each depot. In addition, the VNS algorithm systematically explores the solution space by making local modifications to the initial routes generated by the MARL component, identifying high-quality routes that minimize the total travel distance, energy consumption, and penalty costs. Consequently, by combining RL and VNS, we can benefit from the exploration capabilities of VNS to improve the quality of solutions generated by RL, leading to more efficient and robust routing strategies in dynamic environments. Since both components consider the limited battery capacity of EVs and the availability of charging stations when planning routes, they are able to optimize the routing decisions to ensure that vehicles have sufficient energy to complete their assigned routes and can recharge at appropriate times.

The DMDEVRPTW is defined on a complete directed network $G = (V = C \cup D, E)$, where $C = \{1, 2, \dots, n\}$ is the set of n customers, $D = \{1, 2, \dots, m\}$ represents the set of m depots and $E = \{(i, j) | i, j \in V\}$ is the edge set. The depots act as both the beginning and the end of the vehicle routes, and we assume they also act as charging stations for the electric vehicles. Therefore, we assume that depots inherently provide charging capabilities, meaning that any vehicle returning to its assigned depot is fully charged before leaving it. By this, we attempt to eliminate the need for explicit charging station management. Each customer $i \in C$ has a known demand d_i and a time windows $[a_i, b_i]$ within which the vehicle must arrive at the customer location, and the travel time between customer i and customer j is denoted by t_{ij}. Let K denote a fleet of k vehicles, each having a limited capacity Q and a maximum travel time T within which it must complete a route. The set of routes taken by the vehicles is denoted by W. Moreover, we assume that each vehicle is assigned to one of the depots, and it returns to the depot where it started its route. The dynamic nature of the problem implies that parameters such as demand, service time, travel times, and time windows are subject to change (Ritzinger et al., 2016; Erera et al., 2010; Chang, 2005; Ma et al., 2023). As a result, these parameters may vary over time, and the solution provider cannot know their actual values in advance. To address this situation, the model considers the known variables associated with each customer's status at every time step, focusing on real-time decision-making. The next customer is selected based on the available information at that moment. Upon the vehicle's arrival at a delivery point, the actual values of these parameters are revealed, and this data is used to update both the vehicle's status and the environment.

To be able to apply the RL-based solution, we first model the DMDEVRPTW using a MDP. An MDP model is generally defined by a tuple $M = \langle S, A, R, P, \gamma \rangle$, containing the state space S, an action space A, a reward function $R : S \times A \to R$, a transition probability function $P(s'|s, a)$, and a discount factor γ which can be defined in the range of $[0, 1)$. Within the standard RL framework, the agent observes the state of the environment at each step ($s \in S$) and takes an action ($a \in A$) according to a policy π, based on which it receives a reward, $R(s, a)$, and reaches a new state, s', according to the probability transition function. The goal is to find the policy π that maximizes the long-term reward. The MDP-based DMDEVRPTW model consists of a set of dynamic states, actions, rewards, and transition probabilities defined as follows:

- **States**: The dynamic state s_t at time t represents the current configuration of the system, including the location of the vehicle, the vehicle's remaining battery charge, the location, and demand of each customer, and time windows.
- **Actions**: The action a_t at time t represents the decision made by the autonomous EV, including the selection of the next customer to visit and the route to follow.
- **Rewards**: The reward r_t at time t represents the immediate utility of taking the action a_t in the state s_t, and is defined relative to the cost function and based on total travel distance and the time window violation penalty. The goal is to maximize the total reward (or minimize the total cost) over a finite time horizon T.
- **Transition probabilities**: The transition probability represents the probability of transitioning from one state to another when a certain action is taken and depends on the dynamic parameters of the problem, like the demand and travel times of the customers and the state of the vehicle's battery charge.

4. Core components of the proposed framework

In this section, we outline the key elements that form the foundation of the proposed framework.

4.1. Double deep Q-network

The RL algorithms can generally be classified into two categories: model-free RL, which bases its learning on trial and error, and model-based RL, which attempts to model the environment and select the best possible action according to the learned model. We base our framework on using Q-Learning, which is a model-free, off-policy RL algorithm that learns an action-value function, $Q(s,a)$, the relative benefit of a given action at a given state. The action value function used in Q-Learning is defined as the expected cumulative reward while considering the discount factor:

$$Q_t^{\pi}(s_t,a_t) = E[\Sigma_{i=0}^{\infty}\gamma^i r_{t+k}|s_t = s, a_t = a] \tag{1}$$

The optimal action value function is obtained by selecting the action with the highest Q-value in each state, $Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a)$ which satisfies the Bellman optimal equation:

$$Q^*(s_t,a_t) = E[r_t + \gamma \max_a Q^*(s_{t+1},a)|s_t,a_t] \tag{2}$$

The goal is to maximize the optimal action value function $\max_a Q^*(s,a)$ by finding the optimal policy $\pi^*(s) = \text{argmax}_a Q(s,a)$. The Q-function $Q(s,a)$ represents the expected cumulative reward of taking action a in a state s, and can be updated through value iteration update formulated as follows:

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)) \tag{3}$$

where α is the learning rate, γ is the discount factor, and $\max_a Q(s_{t+1},a)$ is the maximum Q-value over all possible actions in the next state s_{t+1}. The Q-Learning algorithm normally uses a Q-table to store the action values through iterations, which is less efficient when dealing with large-scale state and action spaces. As a result, an approximation function is used to estimate the action value function, minimizing the loss function:

$$L_t(\theta_t) = E[(r_t + \gamma \max_a Q(s_{t+1},a;\theta_t)) - Q(s_t,a_t;\theta_t)]^2 \tag{4}$$

where $(r_t + \gamma \max_a Q(s_{t+1},a;\theta_t))$ is the objective function that is approximated by $Q(s_t,a_t;\theta_t)$ though iterations and iteratively updating θ as follows:

$$\theta_{t+1} = \theta_t + \alpha((r_t + \gamma \max_a Q(s_{t+1},a;\theta_t)) - Q(s_t,a_t;\theta_t))\nabla\theta_t Q(s_t,a_t;\theta_t) \tag{5}$$

The deep Q-network (DQN) uses neural networks in the Q-Learning algorithm to approximate the action value function, making it possible to handle large state space environments (Mnih et al., 2015). In particular, when dealing with large-scale VRPs, numerous customers should be served, which leads to a large state and action space. Therefore, using neural networks can be more efficient than handling a large Q-table. Moreover, being model-free and off-policy makes Q-learning being able to flexibly adapt to different environments. The DQN architecture takes the dynamic state s_t as input at the time t and outputs the Q-values for all possible actions. The network is trained using batches of experiences (s_t,a_t,r_t,s_{t+1}), which are sampled from a replay buffer that stores previous experiences.

Double Q-Learning was initially proposed by hasselt (Hasselt, 2010) to enhance the performance of the Q-Learning algorithm and alleviate its shortcomings, such as overestimation. Similarly, Double Deep Q-network (DDQN) can address the shortcoming of the DQN algorithm (Van Hasselt et al., 2016). The DDQN algorithm uses two neural networks instead of one, namely, a target network and an online network. The target values for the Q-learning update are generated using the target network, which is a duplicate of the online network that is updated less frequently in every N step. DDQN uses the target network for action evaluation, while the optimal course of action in each state is chosen by the online network. In this manner, DDQN enhances stability and performance by lowering the overestimation bias and variance of the Q-learning update. The objective function in DDQN is then defined as

$$r_t + \gamma Q(s_{t+1}, \text{argmax}_a Q(s_{t+1},a;\theta_t);\theta_t^-) \tag{6}$$

where θ_t is the online network parameter and θ_t^- is the target network parameter. In this manner, DDQN can effectively reduce overestimation using the double estimator operation.

To further enhance the performance of the DDQN model, we also make use of prioritized experience relay (PER) buffer (Schaul et al., 2015), which is an improvement over the typical experience replay. PER prioritizes experiences based on their importance, diverging from random sampling by placing emphasis on experiences based on their impact on the agent's behavior, measured through temporal difference error. This prioritization ensures efficient learning and emphasizes the value of experiences that challenge and refine the agent's strategy, stabilizing training, increasing sample efficiency, and finally, enhancing the overall performance of RL systems. The magnitude of the temporal-difference error (TD) between the TD objective and the predicted value is used to calculate the absolute value priority of experiences.

$$p_i = |\delta_i| + e \tag{7}$$

where δ_i is the temporal difference error, p_i is the priority of the ith experience, and e is a small positive offset value.

4.2. Knowledge-guided RL

MARL generalizes MDP to stochastic game (SG). An SG is defined as a tuple $M_{SG} = \langle S, a_1, \ldots, a_n, r_1, \ldots, r_n, P\rangle$ with n agents. Each agent $i \in N$ executes an action a_i at a time t, based on the current system state s_t, transitioning the system to a state s_{t+1} by which each agent receives its corresponding reward $r_i = R_i(s_t,a_t,s_{t+1})$. If all the reward functions associated with agents are equal, it means that all the agents follow the same goal, leading to a cooperative SG (Busoniu et al., 2008).

In order to provide common information among the vehicles, we designed a cooperation module in which all the knowledge of the environment is required to be shared among the agents. Therefore, knowledge such as the service state of the customers and the current joint dynamic state information of the environment is provided to the vehicles and updated over time. Here, we integrate the environment-based regional knowledge into PDDQN, using a property named state–action permissibility (SAP) (Mazumder et al., 2022) that can significantly speed up RL training.

There are two main types of permissibility defined in SAP: Type 1, where an action is permissible only after it has been performed, and Type 2, where an action is permissible without being performed. An action is considered non-permissible if it does not contribute to optimizing the objective or cannot lead to an optimal accumulated reward in the long run.

In this study, we make use of Type 2 SAP, as it can be provided by the domain knowledge of the environment to agents prior to executing the actions. In this regard, we cluster the environment into different regions based on their closeness to each depot, meaning that each depot will have access to a specific region whose customers are closer to that depot. The agents do not have to serve customers who are distant from their initial locations, leading to a decline in the total traveled distance and a lower time complexity of finding a viable solution.

4.3. Variable neighborhood search

The VNS is a meta-heuristic method, initially introduced by Mladenović and Hansen (1997), and is used to find approximate solutions to combinatorial optimization problems. It explores various neighborhood structures, avoiding local optima and aiming towards global optima, using a process named the shaking phase, which works by randomly selecting a solution from the kth neighborhood of the current solution. A local search is also applied to refine the solution if it is not optimal within its neighborhood. Otherwise, if the new solution is better than the current solution, the current solution is replaced with the new one. The VNS algorithm begins with an initial solution and repeats steps such as shaking, local search, and updating iteratively until a stopping

criterion is met, such as a maximum number of iterations without improvement or a time limit (Hansen et al., 2019).

Algorithm 1 illustrates the pseudocode of the general VNS algorithm, starting from an initial solution, modifying it, and exploring its neighborhood to find the best possible solution iteratively until the stopping condition is met. Algorithm 2 shows the shake approach in which the solution is altered randomly, and ultimately, Algorithm 3 illustrates the local search procedure in which the solution is refined by exploring its neighborhood, depending on the operation, which is generally denoted by the *Mutate* phase. In this study, we use the *Two-Opt* operation in the local search to refine the routes traveled by the agents and explore all the possible *Two-Opt* neighborhood of the input solution. Moreover, we also consider the *Two-Opt* operation in the shake procedure, randomly selecting two nodes from the solution and applying the operation based on them.

Algorithm 1: Variable Neighborhood Search Psudocode

Input: a set of neighborhood structures N_l, $l = 1, 2, \ldots, l_{max}$, The Stopping Condition
Output: The best possible solution

```
S = Generate Inital Solution
while The Stopping Condition is not met do
    l = 1
    while l ≤ l_max do
        a) Shaking
        S' = Shake(S, N_l)
        b) Local Search
        S'* = Local Search(S')
        c) Neighborhood Change
        if f(S'*) < f(S) then
            S = S'*
            l = 1
        else
            l = l + 1
        end
    end
end
```

Algorithm 2: Shake Procedure

Input: S, k_{max}
Output: S'

```
for (k = 1; k ≤ k_max; k++) do
    Generate a random neighbor S' of S
    S ← S'
end
```

Algorithm 3: Local Search Procedure

Input: S
Output: S'

```
while The best route is not found do
    S' = Mutate(S)
    if f(S') < f(S) then
        S = S'
    end
end
```

5. The proposed framework

Fig. 2 illustrates the architecture of the proposed framework that integrates knowledge-guided RL, multi-agent deep RL, and variable neighborhood search. As shown in Fig. 2, the main components of the proposed framework include the DRL and VNS modules, as well as a simulator responsible for generating problem instances for the model.

In the following, we describe each module in details.

5.1. Problem simulator

The design of a routing simulator is primarily introduced to provide a reliable environment for DRL, generating numerous VRP instances, as well as managing the alterations made in the environment to which the vehicles interact.

This means that the simulator is responsible for generating DMDE-VRPTW instances for the framework, updating the dynamic state of the environment, and hiding (i.e., masking operation) unavailable customer nodes at each time step, whenever vehicles select a customer and execute an action.

The role of this module is prominent as it enables the DRL model to explore the problem through simulations and learn by executing interactions with the environment, which can be regarded as the basis of RL methods.

5.1.1. Masking operation

There are two types of masking operations considered in this study, including the one responsible for masking the unavailable customer nodes to be selected at the next time step by each agent $i \in N$, and the one which is applied while the DDQN component selects the probability values of all the possible actions for each agent i. The initial type of masking, which is related to masking the previously served customers, is carried out by the simulator component, indicating which customers have been served and which ones should be visited by the vehicle agents in the following time steps. In this respect, all the formerly visited customers are masked for the vehicles, so that they merely focus on the remaining customers at each time step.

5.2. Knowledge-guided MARL

In this framework, we employ a MARL approach with centralized training, decentralized execution, and full cooperation, where the agents are connected to a centralized DDQN. The MARL component includes n homogenous agents ($n \in N$) that cooperate together using a cooperation module. Each agent observes the environment and executes the proper action decided by the DDQN network, receiving a specified reward value. We use the following reward function, in which t_{jivd} denotes the arrival time of $vehicle_{vd}$ to customer j from customer i:

$$\begin{Bmatrix} r_i = -1 & \text{vehicle}_{vd} \text{ is idle}; v \in N; d \in D \\ r_i = 0 & \text{vehicle}_{vd} \text{ returns to depot} \\ r_i = \frac{1}{d_{ij} + \alpha \times (a_j - t_{jivd})} & t_{jivd} < a_j \\ r_i = \frac{1}{d_{ij} + \beta \times (t_{jivd} - b_j)} & t_{jivd} > b_j \end{Bmatrix} \tag{8}$$

At the end of each time step, the cooperation module observes the current state of the environment and updates the agents' information regarding the state of the environment, including the already visited customers and their service status. Therefore, when one agent i visits a customer c, all the other agents will become aware of this matter and the customer c will be masked in the following steps. This is prominent, particularly due to multiple agents being assigned to one depot in real-world applications, leading to higher cooperative execution and shared knowledge among the vehicles.

The application of knowledge-guidance to the DRL component is handled by the use of the second type of masking operation. Integrated in the DRL component's decision-making process, the regional knowledge of the environment reflects the customers that are available for each individual agent $i \in N$, being closer to the vehicle's associated depot. Therefore, each agent is responsible for serving the customers, that are in a specific region to which the depot, and consequently, the

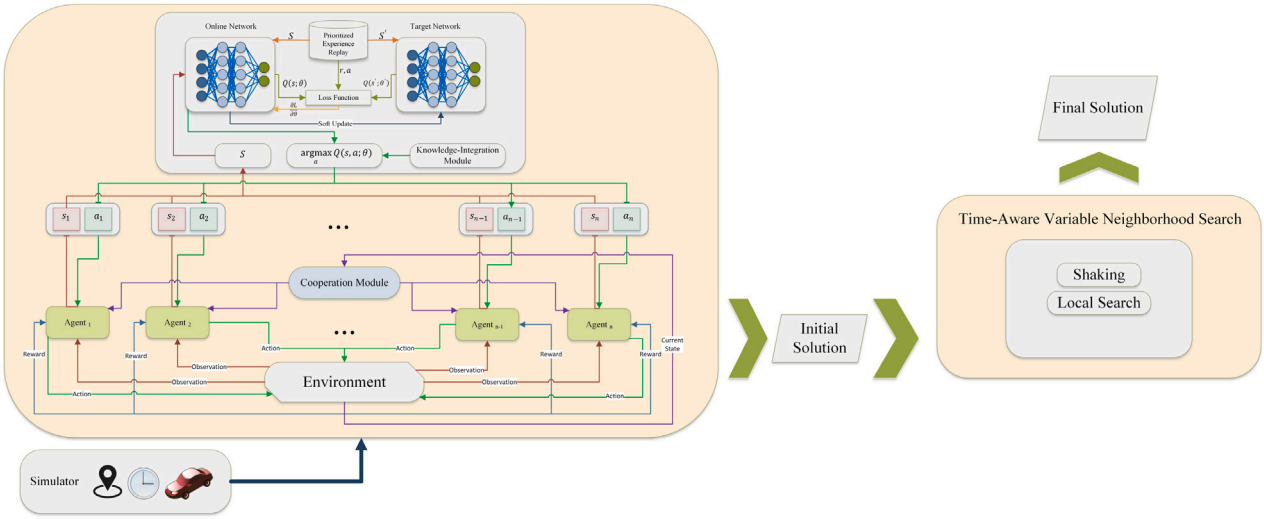


Fig. 2. General architecture of the proposed framework.

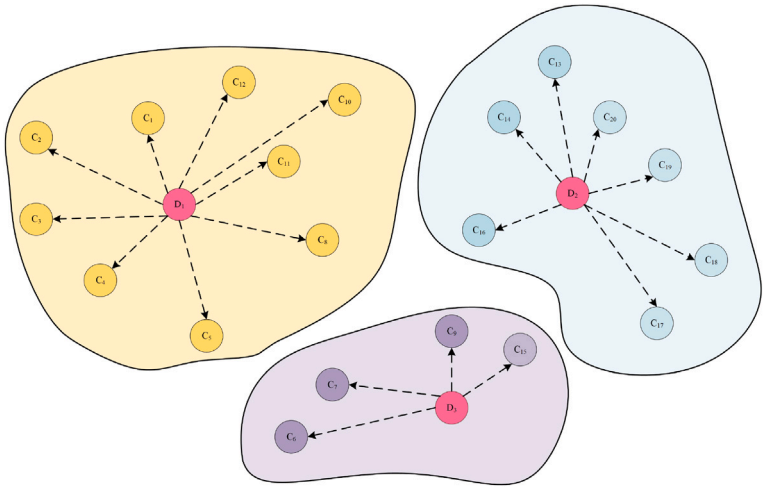


Fig. 3. Associating the customer nodes to their closest depot. Considering a problem instance with C customers, M depots, and N vehicles (each assigned to a depot $d_m; m \in M$), the environment is clustered to different regions (reg_m) whose customers are served by the vehicles ($v_{n,m}$) assigned to their closest depot (d_m).

vehicle is assigned. To clarify this further, Fig. 3 illustrates an instance with $C = 20$ customers, $M = 3$ depots, and N vehicles, in which the customers are assigned to one of the depots according to their closeness to the depot. This approach is inspired by creating the node clusters introduced in Gillett and Johnson (1976), Bezerra et al. (2023), which assign the customers to their closest depot. Algorithm 4 shows the process of assigning a list of closest customers $c \in C$ to each depot $d \in D$.

Once customers are assigned to a depot, the knowledge-guided module within the DDQN component guides the decision-making process for the next action using SAP.

A permissible action is one that leads to the optimal long-term objective, which is minimizing the cost function. Therefore, only actions that contribute to reducing the long-term cost, defined in this study as the total distance and time window violation penalty, are permissible. This means that only customers closer to the vehicles' assigned depots are considered.

In this respect, the regional knowledge of the environment is taken into account and each vehicle is only provided with the list of customers that are located in their specified region. Together with the general masking operation responsible for masking the visited customers, this approach leads the agents (vehicles) to generate the routes with the minimum distance values, while minimizing the time windows violation penalty cost. Moreover, this approach can easily deal with the dynamic version of the problem by assigning the newly arrived customers to their appropriate regions.

5.3. Time-aware VNS

The proposed framework utilizes VNS as a meta-heuristic method to improve the routes generated as the initial solution by the MARL component. The VNS algorithm starts from an initial solution and explores the solution's neighborhoods, aiming towards the global optima. The MARL component, using a knowledge-guided PDDQN model, provides the initial solution for the VNS to re-optimize the constructed routes, using shaking and local search procedures. Numerous operators can be used in the VNS algorithm to explore the neighborhood structure and modify the solution in various way. However, in this study, we use the *Two-Op* operator in both the shaking and the local search phases to refine routes generated in the initial solution.

Once the MARL component generates the routes for each depot $d \in D$ and its assigned vehicles, these routes are updated by applying the VNS algorithm and considering the total traveled distance and the time windows violation penalty cost. In other words, the refined

Algorithm 4: Proposed Depot Clustering Method

Input: C, D
Output: Clusters of Depots-Customers

$C' \xleftarrow{duplicate} C$, $D' \xleftarrow{duplicate} D$
Initiate Clusters $CD[d_i]$ as an empty list for each depot $d_i \in D$
while C' *is not empty* **do**
  $A'[d_i] = \{\}$; Dictionary of closest depot to each customer $c \in C'$; $d_i \in D'$
  $B'[d_i] = \{\}$; Dictionary of second closest depot to each customer $c \in C'$; $d_i \in D'$
  $Prox[c] = \{\}$; Dictionary of the proximity ratio of each customer c to its two closest depots
  $j = 0$
  while $j \leq |C'|$ **do**
    $c = C'[j]$
    $coord_c$ = Coordinates of customer c
    $dist_{cd_i}$ = List of distances from customer c to each depot $d_i \in D'$
    $d_1 = \arg\min_x(dist_{cd_i})$; $d_i \in D'$
    $A'[d_1] \xleftarrow{\cup} c$
    $dist_{cd_i} = dist_{cd_i} \setminus dist_{cd_1}$
    $d_2 = \arg\min_x(dist_{cd_i})$; $d_i \in D' \setminus d_1$
    $B'[d_2] \xleftarrow{\cup} c$
    $Prox[c] = dist_{cd_1}/dist_{cd_2}$
    $j = j + 1$
  end
  Sort $Prox$ in ascending order according to the proximity values $dist_{cd_1}/dist_{cd_2}$
  Assigned Customers $AC = \{\}$
  while $j \leq |C'|$ **do**
    $c = Prox.keys()[j]$
    $d_1 = d_i \quad | \; c \in A'[d_i]$
    $d_2 = d_i \quad | \; c \in B'[d_i]$
    $L_{d_1} = |CD[d_1]|$
    $L_{d_2} = |CD[d_2]|$
    if $L_{d_1} < int(|C|/|D|)$ **then**
      $CD[d_1] \xleftarrow{\cup} c$
      $AC \xleftarrow{\cup} c$
    else if $L_{d_2} < int(|C|/|D|)$ **then**
      $CD[d_2] \xleftarrow{\cup} c$
      $AC \xleftarrow{\cup} c$
    else
      $CD[d_1] \xleftarrow{\cup} c$
      $AC \xleftarrow{\cup} c$
    end
    $j = j + 1$
  end
  $C' \leftarrow C' \setminus AC$
  $j = 0$
  while $j \leq |D'|$ **do**
    $d = D'[j] \quad | \quad d \in D$
    $L_d = |CD[d]|$
    if $L_d \geq int(|C|/|D|)$ **then**
      $D' \leftarrow D' \setminus d$
    end
    $j = j + 1$
  end
end

routes are evaluated in each iteration of the VNS algorithm, ultimately resulting in an optimized route that has the lowest possible total traveled distance, while complying with time-related constraints and minimum time windows violations. Moreover, at each iteration, the newly generated routes are evaluated to make sure that the found solution adheres to the demand constraint. If the solution violates the

demand constraint, the solution is broken at that point, and a new route is formed. Algorithm 5 illustrates how the VNS component of the proposed framework works. Given the route R_d, starting and ending to the depot d, with the associated distance and time-related information, the routes are refined and then compared to the original route based on the following cost:

$$\text{Cost} = \gamma * \text{Distance}(R_d, Distances) + \delta * \text{Time Evaluation}(R_d, TW, ST, \alpha, \beta) \tag{9}$$

where γ and δ denote the weight for the distance and time assessment, respectively. The term Distance$(R_d, Distances)$ calculates the total traveled distance of the given route, and Time Evaluation$(R_d, TW, ST, \alpha, \beta)$ evaluates the route regarding the time constraints, and computes the total time windows violations penalty costs for both the early and ending time windows:

$$\text{Penalty Cost} = \Sigma_{j \in R_d} \alpha * \max(0, a_j - t_{jd}) + \Sigma_{j \in R_d} \beta * \max(0, t_{jd} - b_j) \tag{10}$$

in which α and β denote the penalty cost coefficient of the early and ending time windows penalty costs, respectively, a_i and b_i are the early and ending time windows of the customer $j \in R_d$ which is the jth node of the route R_d generated for depot d, and t_{jd} represents the arrival time of the vehicle v of depot d to the node j in route R_d. We have used $K_{max} = 50$ as stopping condition.

Algorithm 5: Modified Variable Neighborhood Search Algorithm

Input: Route $R_d; d \in D$, $Distances$, Time Windows TW, Service Times ST, Distance Weight γ, Time Weight δ, K_{max}, α, β
Output: The Optimized Route $R_d^{optimized}$

```
k = 1
while k ≤ K_max do
    R'_id = Shake(R_d, k)
    R'*_d = Local Search(R'_d)
    Cost R'*_d = γ * Distance(R'*_d, Distances) + δ *
      Time Evaluation(R'*_d, TW, STα, β)
    Cost R_d = γ * Distance(R_d, Distances) + δ *
      Time Evaluation(R_d, TW, ST, α, β)
    if Cost(R'*_d) < Cost(R_d) then
        R_d = R'*_d
        k = 1
    else
        k = k + 1
    end
end
Return R_d^optimized
```

6. Experimental study

To assess the performance and the reliability of the suggested framework, numerous experiments have been carried out to compare it with established benchmark techniques, in terms of solution quality, cost efficiency, and computational time. We have considered both medium and large-scale problems in our experiments, using instances including 20, 50, 100, and around 1000 customers. This section is structured as outlined below: Section 6.1 describes the baselines and experimental settings used to evaluate the performance of the proposed framework compared to state-of-the-art methods, providing detailed information for each approach. Section 6.2 details the datasets utilized for the experimental study. Sections 6.3 and 6.4 provide detailed analysis of the evaluation findings on deterministic and dynamic environments, considering both randomly generated and real-world benchmarking datasets. The evaluation codes and datasets are publicly available in our Githubrepository.[1]

6.1. Experimental settings

To evaluate the effectiveness of the proposed framework, we compare its performance with that of state-of-the-art VRP solvers introduced by Bezerra et al. (2023) and Wang et al. (2024) on deterministic and dynamic settings, respectively. Since these methods are based on search algorithms, we also compare our proposed framework with two well-known RL-based approaches, namely Actor–Critic (AC) and PDDQN. AC and PDDQN have proven to be effective in both deterministic and dynamic versions of the VRP (Lowe et al., 2017; Schulman et al., 2017; Ahamed et al., 2021; Bdeir et al., 2021). Specifically, Bezerra et al. (2023) address the MDVRPTW using a variable neighborhood search approach, referred to as SGVNSALS. In contrast, Wang et al. (2024) propose an adaptive large neighborhood search (ALNS) method to generate solutions for dynamic VRPs. Therefore, to evaluate the proposed PDDQNVNS, we made use of different benchmark algorithms, suitable either for deterministic or dynamic environments. For deterministic cases, PDDQNVNS was compared against SGVNSALS which is a deterministic algorithm, while also including AC as a baseline. For the dynamic evaluation, we used the ALNS algorithm for dynamic problems as well as PDDQN. Regarding the number of vehicles for each approach, we have considered the minimum required vehicles for routing instances, being equal to the number of depots, in each instance. This is while SGVNSALS considers the number of sub-routs as the number of vehicles required to solve the problem. Due to the dynamic nature of the routing problem, it is assumed that customer demand, travel times, and service times are revealed only when the vehicle reaches a customer or moves along an arc. Moreover, regarding the electric vehicle parameters for our evaluations, we established a simplified energy model to represent vehicle battery dynamics. Each vehicle was initialized with a normalized battery capacity of 1, and the energy consumption is modeled linearly based on the traveled distance and the vehicle's status. We have considered the energy consumption of 0.02 per unit of distance, and a constant energy decay factor of 0.001 being idle. This represents a gradual loss of battery capacity independent of distance traveled. This simplified model allows us to assess the performance of our routing algorithm under controlled and consistent energy constraints, considered for all the following experiments.

The experiments are all executed on a 64-bit quad-core computer with 16 GB of RAM using the Python programming language. The parameters for the experimental methods are as follows:

- The parameters of the SGNVSALS algorithms include *maxlevel* that controls the maximum shake level, *pLS1* and *pLS2* that determine the percentage of maximum iterations allocated to training the algorithm and applying adaptive local search techniques, respectively. The SGNVSALS is executed considering 50 iterations, with a maxlevelf 5, $pLS1 = 0.3$, $pLS2 = 0.3$, $\lambda = 0.6$.
- The ALNS algorithm is executed by considering 30,000 iterations, a start time of 400, and β value of 0.99985, as specified in the referenced scientific paper.
- Regarding the proposed framework, the model was trained considering 200 episodes, $\gamma = 0.99$, batch_size of 64, learning rate of 0.003, initial exploration probability of 0.5, $\alpha = 0.1$, $\beta = 0.2$, distance weight and time weight = 1.
- AC and PDDQN algorithms are executed with network settings alike to the proposed framework, with 200 episodes, $\gamma = 0.99$, batch_size of 64, initial exploration probability of 0.5, and a learning rate of 0.003. PDDQN network utilizes 3 linear layers

[1] https://github.com/ShahbazianR/VRP

Table 2
Characteristics of the benchmark datasets we have used in our experiments.

Dataset	$\|Groups\|$	$\|Instances\|$	$\|C\|$	$\|D\|$
Cordeau	1	20	48–288	4-6
Vidal	2 (a,b)	2×14	360–960	4-12

with dimensionality of 256 and action size, and we have followed the a soft-actor-critic architecture, considering one Actor, two Critic and Value networks for AC, with layer dimensionality of 256.

Despite being challenging, it is important to evaluate the running time of the proposed framework and compare it to state-of-the-art baseline methods. To present a fair comparison among the algorithms, we also have considered the training time of the proposed framework in our evaluation in terms of the run time, reporting the total time required for the proposed framework to generate the solutions. The computation time of each method is calculated considering the whole process of that method on the problem instance, and the results are reported in seconds.

6.2. Test problems

The evaluation of the proposed framework is conducted using both random and well-known routing instances, including numerous random samples with 20, 50 and 100 customers, and the benchmark datasets of Cordeau et al. (2001)[2] and Vidal et al. (2013).[3] The random datasets consist of 500 DMDEVRPTW instances with uniformly generated coordinates, time windows, and service times within the space $[0, 1]$. The demand values are initially uniformly generated within the range $[0, 10]$, considering three different total vehicle capacities based on the number of customers $(20 : 30, 50 : 40, 100 : 50)$, as referenced in Nazari et al. (2018) In addition, the performance of the proposed framework and baseline methods is evaluated across various scenarios, considering different numbers of depots and vehicles. These scenarios range from 2 to 5 depots, with each scenario tested multiple times, by varying the number of vehicles.

Table 2 describes the characteristics of the two well-known benchmark datasets, referred to as *Cordeau* and *Vidal*, used in the experiments. The *Cordeau* benchmark dataset contains one group of 20 instances with 48 to 288 customers and 4 to 6 depots with a capacity from 150 to 200 and route duration of 400 to 500. While, the *Vidal* instances consist of two groups (labeled as a and b) of large scale samples with 360 to 960 customers, 4 to 12 depots, and 4 to 26 vehicles for each depot. Both of the mentioned datasets consider the fleet of vehicles to be homogeneous, and consider both narrow and wide time windows. In order to present results comparable to random instances, we have scaled the datasets to a range of $[0, 1]$. In this respect, the coordinates are transferred to a space of $[0, 100]$ and then converted to $[0, 1]$ by division by 200. A similar manner is applied to demand, service time and time windows values, dividing them by 50, 50, and 1000, which are the maximum values present in the datasets.

6.3. Deterministic evaluation results

Table 3 demonstrates the total distance of the solutions determined by the proposed model, PDDQNVNS, compared to those determined by SGVNSALS and AC on randomly generated routing instances with varying number of customers and depots. The first glance at the findings

Table 3
Comparison of the proposed framework to baseline methods on randomly generated datasets.

Instance	SGVNSALS		AC		PDDQNVNS
	Distance	Gap (%)	Distance	Gap (%)	Distance
2d_2v_20	11.58	44.57	12.78	59.55	**8.01**
2d_2v_50	29.31	96.71	29.62	98.79	**14.90**
2d_2v_100	58.69	186.57	57.88	182.62	**20.48**
3d_3v_20	10.20	45.92	13.19	88.70	**6.99**
3d_3v_50	25.70	86.10	30.44	120.42	**13.81**
3d_3v_100	51.60	144.90	58.95	179.78	**21.07**
4d_4v_20	9.53	45.27	12.82	95.43	**6.56**
4d_4v_50	23.83	84.30	29.89	131.17	**12.93**
4d_4v_100	47.22	133.65	58.74	190.65	**20.21**
5d_5v_20	8.63	41.71	13.38	119.70	**6.09**
5d_5v_50	21.76	81.03	30.77	155.99	**12.02**
5d_5v_100	43.59	123.31	58.59	200.15	**19.52**

reveals a clear performance hierarchy among the compared approaches. As can be seen, the proposed model effectively resulted in higher-quality solutions for all the instances. For example, the total traveled distance for the instance $2d\_2v\_20$ was 8.01 for PDDQNVNS, compared to 11.58 for SGVNSALS and 12.78 for AC. This trend persists with increasing problem size and difficulty, with the proposed model reaching a total distance of 19.52 for $5d\_5v\_100$, in contrast to over two-fold distances of 43.59 and 58.59 by SGVNSALS and AC, respectively. By this, the proposed PDDQNVNS shows effectiveness for providing near-optimal solutions for a wide variety of problem scales, which has serious implications for its applications in several routing scenarios. Comparatively, the divergence between the performance of AC and SGVNSALS against PDDQNVNS rapidly increases with the problem size, which may indicate that the compared approaches may be somehow unable to scale efficiently. Comparing the distances recorded by AC in each group of findings (nd_nv), this algorithm experiences a jump from around 12 to 60 by the increase of problem size from 20 to 100. This shows a tremendous increase in solution cost concerning an increase in routing complexity. PDDQNVNS, on the contrary, maintained its consistency and exhibited a better performance across different instances.

Fig. 4(a) depicts the performance of the proposed model and compares it with SGVNSALS and AC on the real-world *Cordeau* dataset. As we can see, the proposed PDDQNVNS showed a more consistent performance and the lowest total distance across all the cases. Particularly in pr06, pr010, pr015, and pr16, total route distance was notably less than the compared approaches. This shows that PDDQNVNS offers superior performance even for problem with increasing complexity and varied number of vehicles. The ability of the algorithm to cover such low distances highlights its effectiveness in providing high quality solutions and road optimization. SGVNSALS and AC demonstrated more fluctuation and sensitivity to problem size in their performance. In particular, it is clear that AC performed the worse on all problem instances. Although this learning-based approach could eventually learn optimal strategies to a certain degree, the algorithm was far from impressive in achieving a competitive score against other methods. For instance, on problem instance pr20, AC resulted in a total distance of around 120, which was significantly higher than the 77.58 achieved by SGVNSALS and 28.68 by PDDQNVNS. This trend is observed for each instance. Based on the results, the average total distance for AC is 68.27, while SGVNSALS stood at 44.63 and 21.69 for PDDQNVNS. Such extremely higher total distances realized by AC suggest a potential issue of non-coverage and possibly a great generalization from the training data in terms of the *Cordeau* dataset. This indicates that possibly the complexity of this specific VRP instances dataset requires much higher learning mechanisms or, at least, a more appropriate architecture for

[2] Available on: http://www.vrp-rep.org/references/item/cordeau-et-al-2001.html

[3] Available on: http://www.vrp-rep.org/references/item/vidal-et-al-2013.html

the performance of AC. SGVNSALS, on the other hand, showed superior performance compared to AC, highlighting the strength of established meta-heuristics in addressing routing optimization problems. The consistently lower total distances accomplished by SGVNSALS compared to AC draw attention on its efficiency in probing the solution space and finding near-optimal solutions. But, SGVNSALS fell short of the results achieved by the proposed PDDQNVNS approach. On average, SGVNSALS resulted in lower distances compared to AC by about 52% from AC. However, although SGVNSALS outperformed AC, it could not reach the efficacy of the proposed PDDQNVNS, where PDDQNVNS averaged an improvement of 105% over SGVNSALS.

As for *Vidal* dataset (Fig. 4(b)), we can see some shifts in the model performance order. We can see a similar trend for AC in most of the routing instances, which generally was outperformed by the compared approaches. Particularly, in instances pr12a, pr18a, pr19a, pr24a and pr24b, AC produced solutions with much higher total distances than both SGVNSALS and PDDQNVNS. Noteworthy, in pr13a, the performance of AC was comparable to that of SGVNSALS, and a huge deviation from this trend was established in pr14a, pr15a, pr16a, pr14b, pr15b, and pr16b, where AC outperformed SGVNSALS. This indicates that AC could be sensitive to specific problem characteristics or instance structures within the *Vidal* dataset, which may indicate instances where the learning process is more beneficial. Similar to the findings on the *Cordeau* instances, SGVNSALS could outperform AC in most cases, though it fell behind the proposed PDDQNVNS in all instances. Overall, given the mentioned exceptions, total distances recorded by SGVNSALS were lower than that using AC. This can refer to this algorithm being effective in producing in near-optimal routes, as well as its effectiveness to problem size and complexity. On the contrary, the presented model showed high efficiency towards the shortest overall distance across all instances of the *Vidal* routing problems. This might suggest that the model attains a good trade-off between exploration and exploitation, and thereby offers solutions of a better quality. Consequently outperforming the learning-based AC and metaheuristic SGVNSALS, the proposed design of PDDQNVNS turned out to be efficient and effective as it utilized either a better search strategy or represented the problem constraints more effectively.

Overall, the solutions from AC were very much inferior to those produced by SGVNSALS and the proposed PDDQNVNS. Thus, this can raise concerns about the direct applicability of standard RL algorithms into more complex VRP instances, probably because of factors like insufficient exploration, a suboptimal reward function, generalization limitations, and sensitivity to the instance characteristics. The success brought by SGVNSALS and PDDQNVNS suggests that hybrid methods, meaning the methods that use a combination of the learning component with some meta-heuristic approaches, probably hold promise for attaining great improvements in solving complex VRPs.

Table 4 compares the number of vehicles required for the proposed PDDQNVNS with that of SGVNSALS, on the *Cordeau* and *Vidal* real-world routing instances. With respect to the *Cordeau* routing instances, we can see that the proposed model provided high quality solutions with a mere 4 to 6 vehicles, as specified based on the number of depots, for each problem instance. In sharp contrast, SGVNSALS requires much more vehicles from a minimum of 28 vehicles for pr01 to a maximum of 197 for pr19, as this algorithm defines the number of routes equal to the number of required agents. This can reveal a considerable inefficiency in the utilization of vehicles, ultimately leading to higher operational costs and wastage of resources, particularly for large-scale problems. We see a comparable pattern for the *Vidal* instances (from p11a to pr24b), where the proposed approach shows consistently required 4 to 12 vehicles depending on the number of depots and rising with the problem size. Likewise to the *Cordeau* findings, such a pattern shows the ability of PDDQNVNS to cover all the consumers and offer high quality solutions even with much fewer active vehicles. This, therefore, results in much lower vehicle-related expenses, including fuel consumption and hardware maintenance. SGVNSALS, on the other hand, shows an even greater disparity in the number of needed vehicles compared to PDDQNVNS when dealing with noticeably larger routing samples than the *Cordeau* data, requiring vehicles ranging from 225 (in pr17b) to 616 (in pr16a). This pattern supports the notion of inefficiency of SGVNSALS with regard to the vehicle numbers, taking into account that this algorithm equals the number of vehicles with the number of routes in the solution. Such marked difference in vehicle utilization between the two approaches is a clear indication that their operational efficiency is not the same. However, by matching the number of vehicles to the number of depots-the proposed model-is clearly a more streamlined and efficient use of resources. Especially in real-world applications where the minimizing operational costs and maximizing resource utilization are paramount.

6.4. Performance evaluation on dynamic environments

6.4.1. Evaluation results on random datasets

Table 5 reports the obtained results for the proposed PDDQNVNS and ALNS, in terms of the total distance and the execution time (in second) when applied on randomly generated DMDEVRPTW instances. The columns *Distance* and *Run Time* report the total distance and run time (in seconds) obtained by the algorithms, averaged on all instances provided in each specified group. Moreover, to calculate the *Gap* percentage of the algorithm i, we have considered the proposed PDDQNVNS as the base value, and used the following equation:

$$Gap_i = \frac{Distance_i - Distance_{base}}{Distance_{base}} \times 100 \tag{11}$$

The randomly generated dataset includes 500 instances each of both small-scale and medium-scale routing problems, considering 20, 50, and 100 customers for each number of depots and vehicles. Each group of instances are labeled based on the number of depots, vehicles, and customers. For example, *3d_3v_50* refers to a group of 500 instances, including 3 depots, 3 vehicles, and 50 customers. Due to the iterative refinement of the ALNS and SGVNSALS, we have also considered the training time of PDDQNVNS to provide a fair comparison between the algorithms. ALNS is highly dependent to the problem size and could not produce a solution within a reasonable time for 30,000 iterations. Therefore, we limited the number of iterations to 5,000, keeping the runtime manageable.

Table 5 shows that as the number of depots increases, both the total routing distance and the runtime decrease across all the tested methods. The only exception is ALNS, which took more than twice as long when dealing with 5 depots. When it comes to solution quality on the randomly generated datasets, this table highlights that PDDQNVNS consistently outperformed the baseline methods by delivering the best results in terms of total routing distance. As this table shows, PDDQNVNS outperformed the ALNS in 9 out of 12 cases, in terms of total traveled distance by the vehicles, leading to a reduction of 33% in the average overall distance. Fig. 5 depicts and compares the performance of PDDQNVNS (lower bars) to PDDQN (upper bars). The computational results show that the proposed framework consistently outperforms the compared PDDQN baseline in all routing instances, resulting in a reduction in the total distance within a range from 5 to 38. This thereby indicates the effectiveness of the proposed framework in finding solutions with noticeably lower distance, meaning more effectiveness in routing optimization. The examination of the influence of problem size also shows that the total distance of both algorithms increase by the number of customers, as should be expected when there are more locations to visit. However, the relative improvement gained by applying the proposed framework seems to consistently increase with the problem sizes, where larger instances experiencing more reduction in the total distance. For instance, PDDQN yields a distance of 56.72 on the 5d_5v_100 instance, whereas the proposed approach achieves a significantly reduced distance of merely 18.77. On

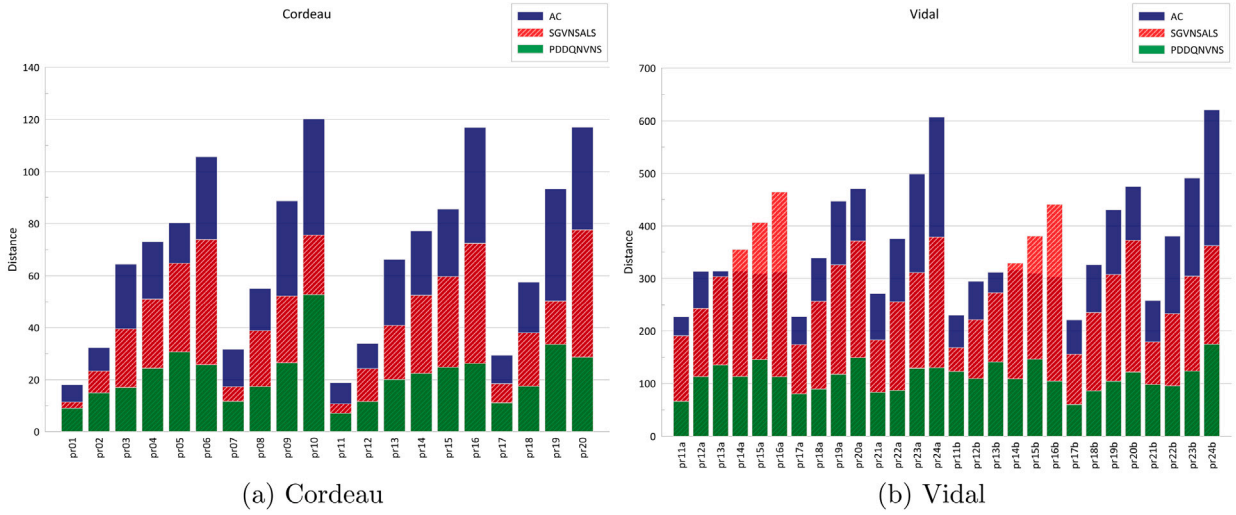


Fig. 4. Comparing the deterministic performance of the proposed model on real-world datasets, *Cordeau* and Vidal.

Table 4
Comparison of the number of vehicles required to serve all customers in SGVNSALS and the PDDQNVNS.

Cordeau			Vidal (a)			Vidal (b)		
Dataset	SGVNSALS	PDDQNVNS	Dataset	SGVNSALS	PDDQNVNS	Dataset	SGVNSALS	PDDQNVNS
pr01	28	4	pr11a	247	4	pr11b	238	4
pr02	62	4	pr12a	323	4	pr12b	306	4
pr03	91	4	pr13a	393	4	pr13b	367	4
pr04	129	4	pr14a	461	4	pr14b	454	4
pr05	161	4	pr15a	537	4	pr15b	518	4
pr06	197	4	pr16a	616	4	pr16b	603	4
pr07	49	6	pr17a	234	6	pr17b	225	6
pr08	103	6	pr18a	354	6	pr18b	337	6
pr09	152	6	pr19a	436	6	pr19b	443	6
pr10	186	6	pr20a	539	6	pr20b	535	6
pr11	30	4	pr21a	284	12	pr21b	282	12
pr12	62	4	pr22a	384	12	pr22b	366	12
pr13	98	4	pr23a	487	12	pr23b	485	12
pr14	134	4	pr24a	602	12	pr24b	587	12
pr15	159	4						
pr16	186	4						
pr17	48	6						
pr18	102	6						
pr19	141	6						
pr20	191	6						

the contrary, in 2d_2v_20 case, the difference is smaller with 12.18 and 7.75 for PDDQN and PDDQNVNS respectively, yet still, PDDQNVNS exhibits an edge over the other. This suggests that PDDQNVNS scales successfully and remains effective for more difficult routing problems.

Comparing the execution time, the proposed framework took longer to execute on randomly generated instances, compared to ALNS. Despite that, PDDQNVNS shows a lower growth in the total running time as the number of customers increases, and maintains a relatively reasonable run time. Moreover, PDDQNVNS shows a relatively shorter run time, compared to ALNS on small-size problems, resulting in faster solutions with a maximum of around 20 s. It should also be noted that the computation time values reported for PDDQNVNS also include the training time, which consumes most of its computation time. This means that once the training process is complete, the approach can produce solutions much faster. Compared to ALNS, the total distance of the solutions on 20 to 100 customers is reduced by around 8% to 25% when PDDQNVNS is applied. In particular, there is a noticeable increase in the gap percentage as the number of customers increases in the experimented instances from 20 to 50. This highlights the robustness of the proposed PDDQNVNS in handling larger-size problems, effectively addressing a critical challenge in real-world applications where the number of customers is often unlimited and frequently grows. The overall changes in the average total traveled distance as the number of customers increases from 20 to 100 demonstrate that the proposed framework achieves the smallest growth, with a maximum increase of 17.69, compared to the ALNS, which shows a significantly higher increase of 39.15. This illustrates that the proposed framework behaves better regarding the scalability property, which is an essential requirement of the vehicle routing solution providers.

Table 6 compares the solution quality and the run time of the proposed PDDQNVNS when the number of vehicles varies. It shows that increasing the number of vehicles, while keeping the number of depots the same (*3d_4v* and *3d_5v* instance groups), results in a modest increase in the average total distance traveled. At the same time, there can be witnessed a minor decrease in the average total computation time. This occurs due to each vehicle needs to visit its assigned depot, which adds to the overall distance traveled, whereas, having more resources to visit the customers only slightly affects the total running time, causing it to decrease slightly. Comparing the situation where the number of depots is increased while keeping the same number

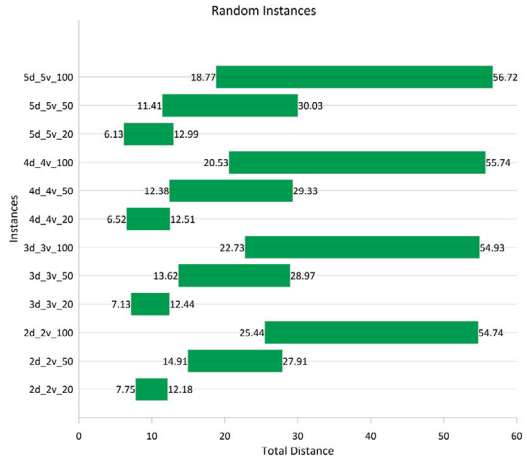


Fig. 5. Comparing the performance of the proposed model to PDDQN. The lower and upper bars refer to PDDQNVNS and PDDQN, respectively.

Table 5
Comparison of the proposed framework to ALNS on randomly generated dataset.

Instance	ALNS			PDDQNVNS	
	Distance	Time (s)	Gap (%)	Distance	Time (s)
2d_2v_20	7.42	35.46	−4.25	7.75	24.91
2d_2v_50	15.99	50.36	7.26	**14.91**	68.96
2d_2v_100	25.40	67.00	−0.16	25.44	146.26
3d_3v_20	7.07	35.67	−0.84	7.13	36.64
3d_3v_50	15.14	52.17	11.14	**13.62**	68.56
3d_3v_100	23.51	74.82	3.44	**22.73**	147.18
4d_4v_20	6.97	35.83	6.90	**6.52**	22.35
4d_4v_50	14.64	56.99	18.24	**12.38**	66.79
4d_4v_100	22.85	79.80	11.32	**20.53**	143.72
5d_5v_20	6.86	36.30	11.90	**6.13**	20.92
5d_5v_50	14.19	57.62	24.36	**11.41**	65.35
5d_5v_100	22.29	84.68	18.74	**18.77**	142.88

Table 6
Comparison of the proposed PDDQNVNS's performance with regard to different vehicle and depot numbers.

Instance	PDDQNVNS	
	Distance	RunTime (s)
3d_4v_20	7.57	23.21
3d_4v_50	14.02	67.83
3d_4v_100	23.13	145.52
3d_5v_20	7.98	22.90
3d_5v_50	14.38	67.04
3d_5v_100	23.59	145.53
4d_5v_20	7.01	23.73
4d_5v_50	13.19	76.17
4d_5v_100	22.54	151.46

of vehicles (*3d_5v* and *4d_5v*), a noticeable decline in the average total distance is witnessed while there is a slight increase in the run time. This is because the region allocation process assigns customers to their closest depots, resulting in more regions as the number of depots increases, which leads to shorter routes.

6.4.2. Evaluation results on benchmark datasets

This section presents the results obtained by the suggested framework on the *Cordeau* and *Vidal* datasets, and compares the proposed framework's performance, in terms of solution quality and running time, with PDDQN and ALNS.

Evaluation results on the Cordeau dataset. The statistical analysis of the results obtained by applying the proposed PDDQNVNS, in comparison to the baseline algorithms on the *Cordeau* benchmark dataset, is illustrated in Table 7 in which the columns $|C|$ and $|D|$ indicate the number of customers and the number of depots for each instance. Fig. 6 depicts the results and the existing gap between PDDQNVNS and the baseline PDDQN. The following experiment includes instances of the *Cordeau* dataset, comprising from 48 to 288 customers, and from 4 to 6 depots. To assess the performance of PDDQNVNS and the PDDQN baseline, we consider the minimum number of vehicles, which is equal to the number of depots.

Table 7 presents a comparative evaluation of ALNS and PDDQNVNS across 20 instances from the *Cordeau* dataset, focusing on total distance, number of vehicles, and runtime. This table underlines that ALNS required, on average, around 79 vehicles to serve all the customers in its generated solutions, while the proposed PDDQNVNS offered the shortest route, requiring only one vehicle per depot. It is noteworthy that we assume the in ALNS the number of vehicles corresponds to the generated sub-routes in the final solution, as it is not explicitly stated in related paper. The ALNS results in routes with a gap value of 57.50% when compared to the solutions determined by the proposed framework. This clearly demonstrates that PDDQNVNS is capable of generating high-quality solutions, which is crucial in real-world scenarios. Specifically, since the total cost of a route is typically influenced by the distance a vehicle travels to reach customers, suboptimal solutions can lead to a significant increase in cost, including factors like fuel consumption, greenhouse emissions, and battery usage. Therefore, the ability to generate high-quality solutions within a reasonable timeframe is of utmost importance. Similarly, Fig. 6 compares the proposed PDDQNVNS with the baseline PDDQN, showing a consistent superior performance by the proposed framework. Although the magnitude of improvement varies across routing instances, depending on the problem size and complexity, we can observe a consistent trend of over 131% reduction in total distance. For instance, the proposed framework could significantly reduce the total traveled distance on pr20, pr10, and pr06 instances, from over 100 to below 50. These instances could represent more complex or challenging routing scenarios where PDDQNVNS's enhanced exploration and exploitation capabilities are particularly beneficial, compared to the PDDQN. By contrast, instances like pr11 and pr01 show relatively smaller improvements, which could be contributed by their smaller problem sizes. Crucially, PDDQNVNS preserves its performance advantage even when the size of the problem increases; in some cases, this absolute difference in distance might increase, thereby showing that the advantage with PDDQNVNS becomes enlarged with increasing problem size. Furthermore, increasing the number of depots typically result in greater total distances for both algorithms, which is logically because the complexity of the routing problem scales with these parameters. Comparing the running times, we can see that PDDQNVNS outperforms ALNS in almost all instances, with a significant gap of over 182%. Furthermore, PDDQNVNS maintains a constant total computation time (also including its training time) relative to the problem size. In contrast, ALNS exhibit varied running times with significant increases, resulting in over 60% more average computation time.

Evaluation results on Vidal dataset. The computational results achieved by the proposed PDDQNVNS, compared to the baseline algorithms on the *Vidal* benchmark datasets (referred to as *a* and *b*), are summarized in Tables 8 and 9. The current evaluation includes both instance groups of *Vidal* benchmark dataset, encompassing both medium-scale and large-scale routing instances with 360 to 960 customers, and 4 to 12 depots. To evaluate the effectiveness of PDDQNVNS, we consider the minimum number of vehicles required, which directly corresponds to the number of depots specified for each instance. Likewise, we have imposed a restriction on the execution of the VNS component in the proposed PDDQNVNS, limiting each execution of VNS to 15 min.

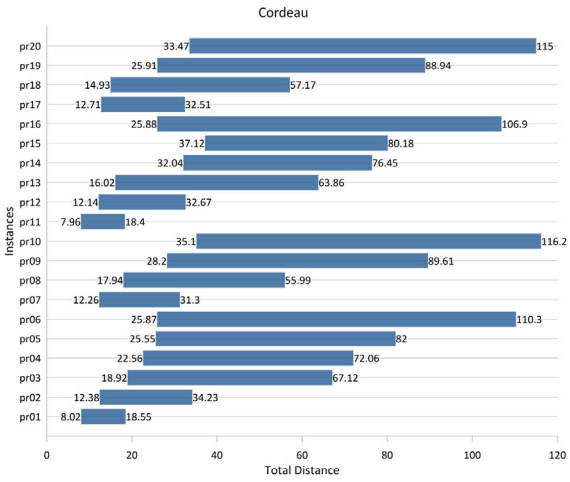


Fig. 6. Comparing the performance of the proposed model to PDDQN on *Cordeau* instances. The lower and upper bars refer to PDDQNVNS and PDDQN, respectively.

Table 7
Comparison of the proposed framework to ALNS on *Cordeau* dataset.

Cordeau	$\|C\|$	$\|D\|$	ALNS			PDDQNVNS		
Instance			Distance	$\|V\|$	Time (s)	Distance	$\|V\|$	Run Time (s)
pr01	48	4	7.60	19	1190.83	8.02	4	168.48
pr02	96	4	19.11	42	2228.56	**12.38**	4	401.42
pr03	144	4	31.32	62	2959.75	**18.92**	4	818.64
pr04	192	4	37.93	95	3269.61	**22.56**	4	1802.29
pr05	240	4	44.37	114	2221.50	**25.55**	4	3627.50
pr06	288	4	51.04	137	6154.87	**25.87**	4	4032.29
pr07	72	6	15.39	31	3152.89	**12.26**	6	230.30
pr08	144	6	26.17	59	1911.45	**17.94**	6	635.91
pr09	216	6	43.54	98	6002.02	**28.20**	6	1269.06
pr10	288	6	57.09	142	6604.08	**35.10**	6	1854.69
pr11	48	4	8.55	14	2865.59	**7.96**	4	173.68
pr12	96	4	19.63	41	2877.52	**12.14**	4	413.17
pr13	144	4	31.44	62	1762.72	**16.02**	4	795.74
pr14	192	4	39.17	90	3388.81	**32.04**	4	869.69
pr15	240	4	46.83	119	5617.36	**37.12**	4	1393.39
pr16	288	4	50.99	133	6071.86	**25.88**	4	3196.00
pr17	72	6	15.02	30	4227.66	**12.71**	6	230.82
pr18	144	6	27.14	66	4990.00	**14.93**	6	670.65
pr19	216	6	42.66	96	2518.38	**25.91**	6	937.83
pr20	288	6	54.43	135	2471.23	**33.47**	6	2139.32
Average			33.47	79.25	3624.33	**21.25**	4.80	1283.04

Table 8 presents a comparison between the results obtained by applying the baseline algorithm and PDDQNVNS on complete instances of the group *a* of *Vidal* dataset. This table highlights that the ALNS used, on average, around 50 times more vehicles to cover all the customers in their generated solutions. On the other hand, PDDQNVNS determines the shortest routes while requiring merely one vehicle per depot and approximately 7 vehicles on average across all the instances. When analyzing the results obtained on the Vidal(a) instances, the ALNS requires on average around 305 vehicles to serve all the customers and determines solutions with an average distance values of 217. However, compared to these algorithms, the proposed framework generates higher quality solutions with significantly lower number of required vehicles. Table 9 illustrates the same comparisons on Vidal(b) routing instances. Similarly, the average number of vehicles used by the baseline algorithm found to be slightly under 51 times more than the specified number of vehicles in the proposed PDDQNVNS. This is while the baseline algorithm required around 302 vehicles, on average, for the Vidal(b) datasets.

Table 8
Comparison of the proposed framework to ALNS on *Vidal* (a) dataset.

Vidal(a)	$\|C\|$	$\|D\|$	ALNS			PDDQNVNS		
Instance			Distance	$\|V\|$	Time (s)	Distance	$\|V\|$	Time (s)
pr11a	360	4	132.69	163	2443.19	**67.56**	4	3737.17
pr12a	480	4	183.03	230	7256.64	**109.60**	4	5166.44
pr13a	600	4	218.40	279	7709.00	**163.55**	4	5472.37
pr14a	720	4	263.71	343	8083.16	**148.96**	4	6491.17
pr15a	840	4	294.95	393	8480.17	**186.28**	4	6555.04
pr16a	960	4	330.15	442	4339.50	**148.20**	4	6810.71
pr17a	360	6	129.46	169	2694.58	**71.07**	6	3001.34
pr18a	520	6	187.71	249	5747.19	**108.78**	6	5375.58
pr19a	700	6	241.19	330	5886.77	**140.23**	6	7285.77
pr20a	880	6	256.92	397	4326.69	**123.32**	6	8656.57
pr21a	420	12	127.93	192	9766.02	**83.14**	12	2596.42
pr22a	600	12	175.55	276	4760.64	**109.02**	12	5023.04
pr23a	780	12	228.02	361	11102.03	**157.59**	12	7594.66
pr24a	960	12	278.59	447	4940.95	**148.96**	12	9874.23
Average			217.74	305.07	6252.61	**126.16**	6.86	5974.32

Fig. 7 compares the proposed PDDQNVNS with the baseline PDDQN, on *Vidal* instances from both groups. The findings demonstrate a similar trend of superior performance by the proposed framework, indicating effectiveness in large-scale route optimization. Importantly, the proposed framework could reduce the total traveled distance to below 150 for all instances, while the compared PDDQN resulted in solutions with a total distance ranging from 220 to over 610. This is particularly important as the large-scale problems would normally require noticeable resource consumption if suboptimal solutions are applied. Both algorithms show variability in performance across different instances, depending on the problem size and complexity. However, the consistent relative performance of PDDQNVNS indicates its robustness, scalability, and adaptability to different problem structures.

Comparing the solution quality, the ALNS produced routes exhibiting a total distance above 72% more than the average value for the suggested PDDQNVNS on both *Vidal* dataset groups, respectively. These findings clearly illustrate the proposed framework's effectiveness in generating superior solutions, compared to the state-of-the-art baseline algorithms. Noticeably, the solutions generated on *Vidal*(b) dataset experienced a slightly lower average total traveled distance on all methods, which may have resulted from the difference in time windows considered in two dataset groups of *a* and *b*. The proposed PDDQNVNS consistently outperforms both baseline methods across all *Vidal* instances, maintaining realistic and reasonable training and computation times proportional to the number of customers. In contrast, the baseline algorithms exhibit significantly increasing computational times as the problem size grows.

6.5. Impact of variable neighborhood search

This section explores the effect of the variable neighborhood component on the overall performance of PDDQNVNS. In this regard, we report the results obtained from the MARL individual component of the proposed framework on *Cordeau* routing instances and compare them with hybrid framework in terms of total traveled distance, which comprises both the MARL and VNS components. Fig. 8(a) shows the total distance values determined by MARL and PDDQNVNS, while Fig. 8(b) displays a notched box plot comparing the two sets of results

As Fig. 8(a) clearly shows, the VNS component resulted in an average reduction of over 20% in the total traveled distance in the majority of *Cordeau* routing instances. Notably, the MARL component itself demonstrates superior performance compared to baseline algorithms. This improvement can be attributed to the region assignment and the knowledge-guided property of our designed MARL component, which considers the closest depots when serving customers. This approach

Table 9
Comparison of the proposed framework to ALNS on *Vidal* (b) dataset.

Vidal(b)	$\|C\|$	$\|D\|$	ALNS			PDDQNVNS		
Instance			Distance	$\|V\|$	Time (s)	Distance	$\|V\|$	Time (s)
pr11b	360	4	125.35	167	2734.19	**59.70**	4	4109.91
pr12b	480	4	155.77	222	7439.75	**76.57**	4	5470.68
pr13b	600	4	195.36	269	5786.86	**94.71**	4	6011.65
pr14b	720	4	249.69	339	3709.62	**146.79**	4	6644.60
pr15b	840	4	291.80	406	5682.55	**108.80**	4	7287.80
pr16b	960	4	300.84	432	9291.50	**103.31**	4	6757.95
pr17b	360	6	118.37	165	3541.98	**63.04**	6	3948.65
pr18b	520	6	167.93	246	8597.22	**93.79**	6	5839.45
pr19b	700	6	217.92	321	9224.67	**144.43**	6	6500.67
pr20b	880	6	251.91	388	4098.66	**123.53**	6	8993.10
pr21b	420	12	126.67	191	9861.22	**83.01**	12	2467.82
pr22b	600	12	172.73	276	10325.28	**126.84**	12	3994.31
pr23b	780	12	231.04	363	11833.41	**142.77**	12	7228.43
pr24b	960	12	278.05	449	5404.69	**138.07**	12	10170.14
Average			205.96	302.43	6966.54	**107.53**	6.86	6101.80

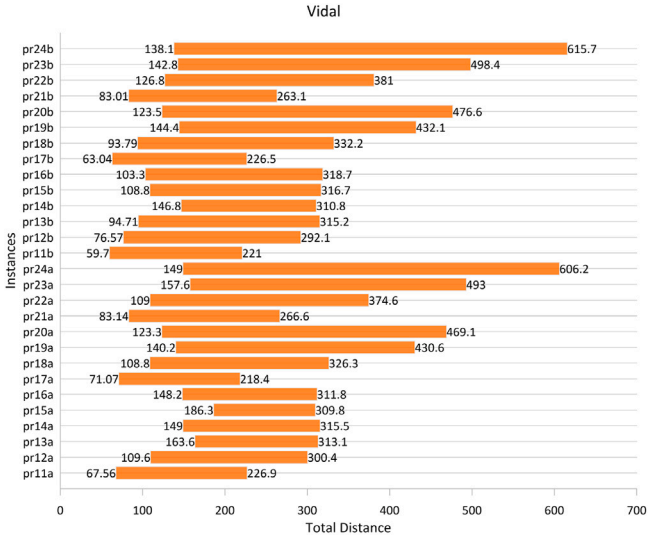


Fig. 7. Comparing the performance of the proposed model to PDDQN on *Vidal* instances. The lower and upper bars refer to PDDQNVNS and PDDQN, respectively.

significantly reduces the total cost of the routing problem. Furthermore, when combined with the VNS module, which refines the solution generated by the MARL, PDDQNVNS demonstrates even superior performance. It decreases the total covered distance to approximately one-third of its value for the MARL component in the majority of *Cordeau* instances.

Fig. 8(b) presents a Notched Box plot comparing MARL and PDDQN-VNS. This plot shows the median, lowest, and highest values for the total traveled distance of both methods. The results indicate that the median total traveled distance using only the MARL component is approximately 34 units, with the distances ranging from 9 to 58 units. However, using the VNS component in the MARL to improve the solutions led to a significant reduction in the median distance covered by more than 14 units. Additionally, the minimum and maximum distance values decreased by approximately 13% and 35%, respectively, across different *Cordeau* routing instances.

Therefore, it can become apparent that the VNS component has the capability to have a significant impact within the proposed PDDQN-VNS, emphasizing that incorporating a blend of learning and heuristic approaches can result in enhanced route planning efficiency, consequently leading to a noticeable reduction in overall costs, which holds great importance in real-world applications. The proposal of optimal results aims to minimize the total distance traveled, while simultaneously adhering to time windows constraints, along with restricting the emission of harmful greenhouse gases and minimizing electricity usage, which are crucial factors in addressing routing challenges within the domain of electric vehicles.

7. Conclusions and future works

In this paper, we studied the dynamic multi-depot vehicle routing problem with time windows for autonomous electric vehicles. The problem under study has been modeled as a Markov Decision Process, with the aim of employing reinforcement learning techniques for its solution. In particular, we have proposed a hybrid knowledge-guided multi-agent RL-based framework, which uses a variable neighborhood search VNS, to further enhance the final solution determined. We performed extensive evaluations using both randomly generated and benchmark datasets, namely *Cordeau* and Vidal, and compared our proposed framework against two state-of-the-art algorithms. As the obtained findings illustrate, our proposed framework shown superior performance in terms of both solution quality and scalability compared to the benchmark algorithms when considering different numbers of clients. In comparison to the state-of-the-art vehicle routing algorithms, PDDQNVNS enhanced the average total route distance by a maximum of over 70% when compared to the experimental baselines on VRP *Cordeau* instances. Furthermore, the results on the complete medium- and large-scale instances illustrates the effectiveness of our proposed framework, along with its ability to easily scale to large-scale problems. According to the obtained results, the proposed framework is able to lead to a significant reduction in both total traveled distance and computation time, when compared with the state-of-the-art baselines on large-scale problems. Therefore, the experiment results show that the proposed algorithms are promising in terms of discovering solutions with lower energy usage and the time necessary to complete customer service.

As for the future directions, more advanced RL-based techniques and integration with attention mechanisms, such as using multi-head attention or encoder–decoder, can effectively enhance the performance of the proposed framework. Moreover, more complex approaches, such as considering context-embeddings, can also be utilized to further assist the framework in capturing aspects of the problem.

CRediT authorship contribution statement

Reza Shahbazian: Writing – original draft, Validation, Methodology, Investigation, Conceptualization. **Alessia Ciacco:** Methodology, Investigation, Conceptualization. **Giusy Macrina:** Methodology, Investigation, Conceptualization. **Francesca Guerriero:** Writing – review & editing, Supervision, Methodology, Investigation, Funding acquisition, Conceptualization.

Acknowledgments

This research was supported by a grant awarded by the “European Union – Next Generation EU” under the “PRIN 2022” project: SMOTION (ID 2022EAECWJ), CUP H53D23002000006.

Data availability

Both the generated and publicly available datasets are accessible online in our GithubRepository.

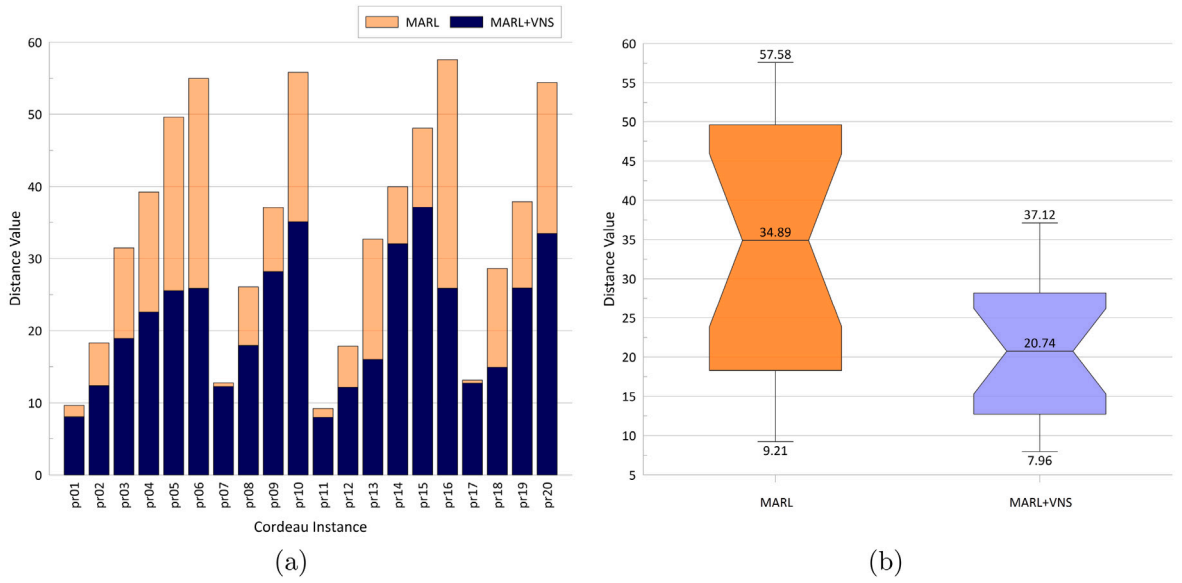


Fig. 8. Comparing the effect of the VNS component on the performance of proposed PDDQNVNS.

References

Ahamed, T., Zou, B., Farazi, N.P., Tulabandhula, T., 2021. Deep reinforcement learning for crowdsourced urban delivery. Transp. Res. Part B: Methodol. 152, 227–257.

Ajao, Q., Oludamilare, O., 2023. Safety challenges and analysis of autonomous electric vehicle development: Insights from on-road testing and accident reports. arXiv preprint arXiv:2305.12665.

Alcaraz, J.J., Losilla, F., Caballero-Arnaldos, L., 2022. Online model-based reinforcement learning for decision-making in long distance routes. Transp. Res. Part E: Logist. Transp. Rev. 164, 102790.

Ammar, A.B., Loukil, T.M., Cheikh, M., 2022. A survey on the hybrid vehicle routing problem: Literature review. In: 2022 14th International Colloquium of Logistics and Supply Chain Management. LOGISTIQUA, IEEE, pp. 1–6.

Bai, R., Chen, X., Chen, Z.-L., Cui, T., Gong, S., He, W., Jiang, X., Jin, H., Jin, J., Kendall, G., et al., 2023. Analytics and machine learning in vehicle routing research. Int. J. Prod. Res. 61 (1), 4–30.

Basso, R., Kulcsár, B., Sanchez-Diaz, I., 2021. Electric vehicle routing problem with machine learning for energy prediction. Transp. Res. Part B: Methodol. 145, 24–55. http://dx.doi.org/10.1016/j.trb.2020.12.007, URL https://www.sciencedirect.com/science/article/pii/S0191261520304549.

Baty, L., Jungel, K., Klein, P.S., Parmentier, A., Schiffer, M., 2024. Combinatorial optimization-enriched machine learning to solve the dynamic vehicle routing problem with time windows. Transp. Sci..

Bdeir, A., Boeder, S., Dernedde, T., Tkachuk, K., Falkner, J.K., Schmidt-Thieme, L., 2021. RP-DQN: An application of Q-learning to vehicle routing problems. In: KI 2021: Advances in Artificial Intelligence: 44th German Conference on AI, Virtual Event, September 27–October 1, 2021, Proceedings 44. Springer, pp. 3–16.

Bello, I., Pham, H., Le, Q.V., Norouzi, M., Bengio, S., 2016. Neural combinatorial optimization with reinforcement learning. arXiv preprint arXiv:1611.09940.

Bezerra, S.N., Souza, M.J.F., de Souza, S.R., 2023. 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. 149, 106016.

Bogyrbayeva, A., Meraliyev, M., Mustakhov, T., Dauletbayev, B., 2024. Machine learning to solve vehicle routing problems: A survey. IEEE Trans. Intell. Transp. Syst. 25 (6), 4754–4772.

Bogyrbayeva, A., Yoon, T., Ko, H., Lim, S., Yun, H., Kwon, C., 2023. A deep reinforcement learning approach for solving the traveling salesman problem with drone. Transp. Res. Part C: Emerg. Technol. 148, 103981.

Bouanane, K., Amrani, M.E., Benadada, Y., 2022. The vehicle routing problem with simultaneous delivery and pickup: a taxonomic survey. Int. J. Logist. Syst. Manag. 41 (1–2), 77–119.

Busoniu, L., Babuska, R., De Schutter, B., 2008. A comprehensive survey of multiagent reinforcement learning. IEEE Trans. Syst. Man Cybern. Part C (Appl. Reviews) 38 (2), 156–172.

Caceres-Cruz, J., Arias, P., Guimarans, D., Riera, D., Juan, A.A., 2014. Rich vehicle routing problem: Survey. ACM Comput. Surv. 47 (2), 1–28.

Chang, M.-S., 2005. A vehicle routing problem with time windows and stochastic demands. J. Chin. Inst. Eng. 28 (5), 783–794.

Chen, J., Zong, Z., Zhuang, Y., Yan, H., Jin, D., Li, Y., 2023. Reinforcement learning for practical express systems with mixed deliveries and pickups. ACM Trans. Knowl. Discov. from Data 17 (3), 1–19.

Cordeau, J.-F., Laporte, G., Mercier, A., 2001. A unified tabu search heuristic for vehicle routing problems with time windows. J. Oper. Res. Soc. 52 (8), 928–936.

Dantzig, G.B., Ramser, J.H., 1959. The truck dispatching problem. Manag. Sci. 6 (1), 80–91.

Dubey, N., Tanksale, A., 2023. A multi-depot vehicle routing problem with time windows, split pickup and split delivery for surplus food recovery and redistribution. Expert Syst. Appl. 232, 120807.

Erera, A.L., Morales, J.C., Savelsbergh, M., 2010. The vehicle routing problem with stochastic demand and duration constraints. Transp. Sci. 44 (4), 474–492.

Eßer, J., Bach, N., Jestel, C., Urbann, O., Kerner, S., 2022. Guided reinforcement learning: A review and evaluation for efficient and effective real-world robotics [survey]. IEEE Robot. Autom. Mag. 30 (2), 67–85.

Gillett, B.E., Johnson, J.G., 1976. Multi-terminal vehicle-dispatch algorithm. Omega 4 (6), 711–718.

Haider, Z., Charkhgard, H., Kim, S.W., Kwon, C., 2019. Optimizing the relocation operations of free-floating electric vehicle sharing systems. Available At SSRN 3480725.

Hansen, P., Mladenović, N., Brimberg, J., Pérez, J.A.M., 2019. Variable Neighborhood Search. Springer.

Hasselt, H., 2010. Double Q-learning. Adv. Neural Inf. Process. Syst. 23.

He, M., Wei, Z., Wu, X., Peng, Y., 2021. An adaptive variable neighborhood search ant colony algorithm for vehicle routing problem with soft time windows. IEEE Access 9, 21258–21266.

Hildebrandt, F.D., Thomas, B.W., Ulmer, M.W., 2023. Opportunities for reinforcement learning in stochastic dynamic vehicle routing. Comput. Oper. Res. 150, 106071.

Hottung, A., Tierney, K., 2022. Neural large neighborhood search for routing problems. Artificial Intelligence 313, 103786. http://dx.doi.org/10.1016/j.artint.2022.103786, URL https://www.sciencedirect.com/science/article/pii/S0004370222001266.

Iklassov, Z., Sobirov, I., Solozabal, R., Takáč, M., 2023. Reinforcement learning approach to stochastic vehicle routing problem with correlated demands. IEEE Access 11, 87958–87969. http://dx.doi.org/10.1109/ACCESS.2023.3306076.

James, J., Yu, W., Gu, J., 2019. Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. IEEE Trans. Intell. Transp. Syst. 20 (10), 3806–3817.

Kalakanti, A.K., Verma, S., Paul, T., Yoshida, T., 2019. RL SolVeR pro: Reinforcement learning for solving vehicle routing problem. In: 2019 1st International Conference on Artificial Intelligence and Data Sciences (AiDAS). pp. 94–99. http://dx.doi.org/10.1109/AiDAS47888.2019.8970890.

Karpatne, A., Atluri, G., Faghmous, J.H., Steinbach, M., Banerjee, A., Ganguly, A., Shekhar, S., Samatova, N., Kumar, V., 2017. Theory-guided data science: A new paradigm for scientific discovery from data. IEEE Trans. Knowl. Data Eng. 29 (10), 2318–2331.

Keskin, M., Çatay, B., Laporte, G., 2021. A simulation-based heuristic for the electric vehicle routing problem with time windows and stochastic waiting times at recharging stations. Comput. Oper. Res. 125, 105060.
Kim, G., Ong, Y.-S., Heng, C.K., Tan, P.S., Zhang, N.A., 2015. City vehicle routing problem (city VRP): A review. IEEE Trans. Intell. Transp. Syst. 16 (4), 1654–1666.
Kopelias, P., Demiridi, E., Vogiatzis, K., Skabardonis, A., Zafiropoulou, V., 2020. Connected & autonomous vehicles–environmental impacts–a review. Sci. Total Environ. 712, 135237.
Kucukoglu, I., Dewil, R., Cattrysse, D., 2021. The electric vehicle routing problem and its variations: A literature review. Comput. Ind. Eng. 161, 107650.
Laporte, G., Semet, F., 2002. Classical heuristics for the capacitated VRP. In: The Vehicle Routing Problem. SIAM, pp. 109–128.
Lei, K., Guo, P., Wang, Y., Wu, X., Zhao, W., 2022. Solve routing problems with a residual edge-graph attention neural network. Neurocomputing 508, 79–98.
Lenstra, J.K., Kan, A.R., 1981. Complexity of vehicle routing and scheduling problems. Networks 11 (2), 221–227.
Li, Z., Chen, Q., Koltun, V., 2018. Combinatorial optimization with graph convolutional networks and guided tree search. Adv. Neural Inf. Process. Syst. 31.
Li, J., Ma, Y., Gao, R., Cao, Z., Lim, A., Song, W., Zhang, J., 2022. Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem. IEEE Trans. Cybern. 52 (12), 13572–13585. http://dx.doi.org/10.1109/TCYB.2021.3111082.
Lin, B., Ghaddar, B., Nathwani, J., 2021. Deep reinforcement learning for the electric vehicle routing problem with time windows. IEEE Trans. Intell. Transp. Syst. 23 (8), 11528–11538.
Lin, B., Ghaddar, B., Nathwani, J., 2022. Deep reinforcement learning for the electric vehicle routing problem with time windows. IEEE Trans. Intell. Transp. Syst. 23 (8), 11528–11538. http://dx.doi.org/10.1109/TITS.2021.3105232.
Liu, F., Lu, C., Gui, L., Zhang, Q., Tong, X., Yuan, M., 2023. Heuristics for vehicle routing problem: A survey and recent advances. arXiv preprint arXiv:2303.04147.
Lowe, R., Wu, Y.I., Tamar, A., Harb, J., Pieter Abbeel, O., Mordatch, I., 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. Adv. Neural Inf. Process. Syst. 30.
Ma, W., Zeng, L., An, K., 2023. Dynamic vehicle routing problem for flexible buses considering stochastic requests. Transp. Res. Part C: Emerg. Technol. 148, 104030.
Macrina, G., Di Puglia Pugliese, L., Guerriero, F., Laganà, D., 2017. The vehicle routing problem with occasional drivers and time windows. In: Optimization and Decision Science: Methodologies and Applications: ODS, Sorrento, Italy, September 4-7, 2017 47. Springer, pp. 577–587.
Macrina, G., Laporte, G., Guerriero, F., Pugliese, L.D., 2019a. An energy-efficient green-vehicle routing problem with mixed vehicle fleet, partial battery recharging and time windows. European J. Oper. Res. 276 (3), 971–982.
Macrina, G., Pugliese, L.D., Guerriero, F., Laporte, G., 2019b. The green mixed fleet vehicle routing problem with partial battery recharging and time windows. Comput. Oper. Res. 101, 183–199.
Mazumder, S., Liu, B., Wang, S., Zhu, Y., Liu, L., Li, J., 2018. Action permissibility in deep reinforcement learning and application to autonomous driving. KDD' 18 Deep. Learn. Day.
Mazumder, S., Liu, B., Wang, S., Zhu, Y., Yin, X., Liu, L., Li, J., 2022. Knowledge-guided exploration in deep reinforcement learning. arXiv preprint arXiv:2210.15670.
Mladenović, N., Hansen, P., 1997. Variable neighborhood search. Comput. Oper. Res. 24 (11), 1097–1100.
Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al., 2015. Human-level control through deep reinforcement learning. Nature 518 (7540), 529–533.
Nazari, M., Oroojlooy, A., Snyder, L., Takác, M., 2018. Reinforcement learning for solving the vehicle routing problem. Adv. Neural Inf. Process. Syst. 31.
Pan, W., Liu, S.Q., 2023. Deep reinforcement learning for the dynamic and uncertain vehicle routing problem. Appl. Intell. 53 (1), 405–422.
Phiboonbanakit, T., Horanont, T., Huynh, V.-N., Supnithi, T., 2021. A hybrid reinforcement learning-based model for the vehicle routing problem in transportation logistics. IEEE Access 9, 163325–163347. http://dx.doi.org/10.1109/ACCESS.2021.3131799.
Qin, W., Zhuang, Z., Huang, Z., Huang, H., 2021. A novel reinforcement learning-based hyper-heuristic for heterogeneous vehicle routing problem. Comput. Ind. Eng. 156, 107252. http://dx.doi.org/10.1016/j.cie.2021.107252, URL https://www.sciencedirect.com/science/article/pii/S036083522100156X.
Quirion-Blais, O., Chen, L., 2021. A case-based reasoning approach to solve the vehicle routing problem with time windows and drivers' experience. Omega 102, 102340.
Raja, G., Saravanan, G., Prathiba, S.B., Akhtar, Z., Khowaja, S.A., Dev, K., 2023. Smart navigation and energy management framework for autonomous electric vehicles in complex environments. IEEE Internet Things J..
Ren, L., Fan, X., Cui, J., Shen, Z., Lv, Y., Xiong, G., 2022. A multi-agent reinforcement learning method with route recorders for vehicle routing in supply chain management. IEEE Trans. Intell. Transp. Syst. 23 (9), 16410–16420.
Ritzinger, U., Puchinger, J., Hartl, R.F., 2016. A survey on dynamic and stochastic vehicle routing problems. Int. J. Prod. Res. 54 (1), 215–231.
Sar, K., Ghadimi, P., 2023. A systematic literature review of the vehicle routing problem in reverse logistics operations. Comput. Ind. Eng. 109011.
Schaul, T., Quan, J., Antonoglou, I., Silver, D., 2015. Prioritized experience replay. arXiv preprint arXiv:1511.05952.
Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.
Shahbazian, R., Pugliese, L.D., Guerriero, F., Macrina, G., 2024. Integrating machine learning into vehicle routing problem: Methods and applications. IEEE Access 12, 93087–93115.
da Silva Junior, O.S., Leal, J.E., Reimann, M., 2021. A multiple ant colony system with random variable neighborhood descent for the dynamic vehicle routing problem with time windows. Soft Comput. 25 (4), 2935–2948.
Sluijk, N., Florio, A.M., Kinable, J., Dellaert, N., Van Woensel, T., 2023. Two-echelon vehicle routing problems: A literature review. European J. Oper. Res. 304 (3), 865–886.
Stodola, P., Nohel, J., 2022. Adaptive ant colony optimization with node clustering for the multi-depot vehicle routing problem. IEEE Trans. Evol. Comput..
Tan, F., Chai, Z.-y., Li, Y.-l., 2023. Multi-objective evolutionary algorithm for vehicle routing problem with time window under uncertainty. Evol. Intell. 16 (2), 493–508.
Van Hasselt, H., Guez, A., Silver, D., 2016. Deep reinforcement learning with double q-learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30.
Vidal, T., Crainic, T.G., Gendreau, M., Prins, C., 2013. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. Comput. Oper. Res. 40 (1), 475–489.
Wang, Y., Qiu, D., He, Y., Zhou, Q., Strbac, G., 2023a. Multi-agent reinforcement learning for electric vehicle decarbonized routing and scheduling. Energy 284, 129335. http://dx.doi.org/10.1016/j.energy.2023.129335, URL https://www.sciencedirect.com/science/article/pii/S0360544223027299.
Wang, S., Sun, W., Huang, M., 2024. An adaptive large neighborhood search for the multi-depot dynamic vehicle routing problem with time windows. Comput. Ind. Eng. 110122.
Wang, Y., Wei, Y., Wang, X., Wang, Z., Wang, H., 2023b. A clustering-based extended genetic algorithm for the multidepot vehicle routing problem with time windows and three-dimensional loading constraints. Appl. Soft Comput. 133, 109922.
Willard, J., Jia, X., Xu, S., Steinbach, M., Kumar, V., 2022. Integrating scientific knowledge with machine learning for engineering and environmental systems. ACM Comput. Surv. 55 (4), 1–37.
Xu, Y., Fang, M., Chen, L., Xu, G., Du, Y., Zhang, C., 2022. Reinforcement learning with multiple relational attention for solving vehicle routing problems. IEEE Trans. Cybern. 52 (10), 11107–11120. http://dx.doi.org/10.1109/TCYB.2021.3089179.
Zhang, S., Chen, M., Zhang, W., Zhuang, X., 2020a. Fuzzy optimization model for electric vehicle routing problem with time windows and recharging stations. Expert Syst. Appl. 145, 113123. http://dx.doi.org/10.1016/j.eswa.2019.113123, URL https://www.sciencedirect.com/science/article/pii/S0957417419308401.
Zhang, K., He, F., Zhang, Z., Lin, X., Li, M., 2020b. Multi-vehicle routing problems with soft time windows: A multi-agent reinforcement learning approach. Transp. Res. Part C: Emerg. Technol. 121, 102861.
Zhang, J., Luo, K., Florio, A.M., Van Woensel, T., 2023. Solving large-scale dynamic vehicle routing problems with stochastic requests. European J. Oper. Res. 306 (2), 596–614.
Zhao, J., Mao, M., Zhao, X., Zou, J., 2020. A hybrid of deep reinforcement learning and local search for the vehicle routing problems. IEEE Trans. Intell. Transp. Syst. 22 (11), 7208–7218.
Zhu, Z., Tang, X., Qin, Y., Huang, Y., Hashemi, E., 2023. A survey of lateral stability criterion and control application for autonomous vehicles. IEEE Trans. Intell. Transp. Syst..
Zirour, M., 2008. Vehicle routing problem: models and solutions. J. Qual. Meas. Anal. JQMA 4 (1), 205–218.
Zong, Z., Xia, T., Zheng, M., Li, Y., 2024. Reinforcement learning for solving multiple vehicle routing problem with time window. ACM Trans. Intell. Syst. Technol..