European Journal of Operational Research xxx (xxxx) xxx

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Innovative Applications of O.R.

# Deep reinforcement learning for solving the stochastic e-waste collection problem

Dang Viet Anh Nguyen [a], Aldy Gunawan [a,*], Mustafa Misir [b], Lim Kwan Hui [c], Pieter Vansteenwegen [d]

[a] School of Computing and Information Systems, Singapore Management University, 80 Stamford Road, Singapore, 178902, Singapore
[b] Division of Natural and Applied Sciences, Duke Kunshan University, Duke Avenue No: 8, Kunshan, Jiangsu, 215316, China
[c] Information Systems Technology and Design Pillar, Singapore University of Technology and Design, 8 Somapah Road, Singapore, 487372, Singapore
[d] Institute for Mobility (CIB), KU Leuven, Celestijnenlaan 300 - box 2422, 3001 Leuven, Belgium

ARTICLE INFO

*Keywords:*
Vehicle routing problem
Deep reinforcement learning
E-waste
Adaptive operator selection

ABSTRACT

With the growing influence of the internet and information technology, Electrical and Electronic Equipment (EEE) has become a gateway to technological innovations. However, discarded devices, also called e-waste, pose a significant threat to the environment and human health if not properly treated, disposed of, or recycled. In this study, we extend a novel model for the e-waste collection in an urban context: the Heterogeneous VRP with Multiple Time Windows and Stochastic Travel Times (HVRP-MTWSTT). We propose a solution method that employs deep reinforcement learning to guide local search heuristics (DRL-LSH). The contributions of this paper are as follows: (1) HVRP-MTWSTT represents the first stochastic VRP in the context of the e-waste collection problem, incorporating complex constraints such as multiple time windows across a multi-period horizon with a heterogeneous vehicle fleet, (2) The DRL-LSH model uses deep reinforcement learning to provide an online adaptive operator selection layer, selecting the appropriate heuristic based on the search state. The computational experiments demonstrate that DRL-LSH outperforms the state-of-the-art hyperheuristic method by 24.26% on large-scale benchmark instances, with the performance gap increasing as the problem size grows. Additionally, to demonstrate the capability of DRL-LSH in addressing real-world problems, we tested and compared it with reference metaheuristic and hyperheuristic algorithms using a real-world e-waste collection case study in Singapore. The results showed that DRL-LSH significantly outperformed the reference algorithms on a real-world instance in terms of operating profit.

## 1. Introduction

Under the growing influence of the internet and information technology, Electrical and Electronic Equipment (EEE) has evolved into a gateway for accessing technological innovations. The rapid advancements in both hardware and software have significantly shortened the update and release cycles of EEE, which offers convenience and comfort in our daily lives. However, from an environmental standpoint, the rapid turnover of EEE imposes substantial pressure on the environment due to their shortened life cycles. E-waste, or Waste Electrical and Electronic Equipment (WEEE), encompasses electrical and electronic devices that have reached the end of their functionality, service life, and life cycle, including all components, sub-assemblies, and consumables present at the time of disposal (Pérez-Belis et al., 2015). Improper disposal can lead to environmental pollution, posing risks to human health and squandering valuable raw materials that could be extracted and processed for new products. Among waste categories, e-waste has emerged as one of the fastest-growing waste streams (Chaudhary & Vrat, 2018). It is estimated that approximately 61.3 million tons of e-waste will be discarded in 2023 (WEEE-Forum, 2023). However, only 17.4% of the total global e-waste is collected and recycled in an environmentally friendly manner. The cost of waste collection and transportation typically accounts for 60%–80% of the total waste management cost (Rangga et al., 2019). Therefore, effective transportation management in e-waste collection can significantly reduce operational costs. Additionally, the collection process is typically carried out by a heavy vehicle fleet. Without adequate planning, this can lead to numerous issues in urban areas, including increased pressure on road networks and worsened urban air pollution.

Vehicle Routing Problems (VRPs) have been extensively studied in the context of waste recycling transportation and closed-loop supply

* Corresponding author.
*E-mail address:* aldygunawan@smu.edu.sg (A. Gunawan).

https://doi.org/10.1016/j.ejor.2025.04.033
Received 9 December 2023; Accepted 19 April 2025
Available online 4 May 2025

chains (Nowakowski et al., 2018; Pourhejazy et al., 2021). In the context of e-waste collection, the variant of VRP typically involves complex planning tasks over a multi-period horizon, encompassing both fixed collection points (e-bins) and dynamic customer requests for household collection. These customer requests, often associated with time windows that change daily, introduce dynamic aspects to the problem (Gunawan, Nguyen , Yu et al., 2023; Pourhejazy et al., 2021; Szwarc et al., 2021).

Although prior research has addressed these practical considerations, significant limitations persist in modeling e-waste collection systems — chiefly, the assumption of deterministic travel times. This assumption directly impacts service reliability, operational efficiency, and the cost-effectiveness of e-waste collection, particularly in daily collection tasks. Such tasks are highly sensitive to time-dependent edge traversal durations and vehicle speeds (Gendreau et al., 2015). E-waste collection tasks are typically executed by a heterogeneous fleet of vehicles, constrained by operational factors such as e-waste size variation and road-width restrictions. The use of different vehicle types also introduces variability in speed, which significantly affects travel times. For example, larger vehicles may move slower due to traffic or require alternative routes to avoid narrow roads, whereas smaller vehicles may navigate faster but face random delays in dense traffic zones. Moreover, strategic placement of e-bins in high-traffic locations, such as malls and schools, makes travel times susceptible to local congestion. Like other urban routing problems, e-waste collection involves navigating densely populated areas where travel time stochasticity fluctuates significantly based on factors like time of day, road width, and congestion patterns. From an enterprise perspective, accounting for stochastic travel times can enhance service reliability, ultimately leading to higher customer satisfaction. Therefore, to make e-waste collection systems more realistic and effective, it is necessary to model the problem as a variant of the stochastic VRP.

Developing solution methods for stochastic variants of the e-waste collection problem presents substantial challenges. State-of-the-art approaches solving VRPs often model the problem as a sequential decision-making task, tackled either through multi-stage optimization or reinforcement learning (RL) (Baty et al., 2024). However, both approaches face significant drawbacks in the context of stochastic e-waste collection. Multi-stage optimization suffers from high computational complexity, scalability issues, and limited adaptability to real-time updates. It also depends heavily on accurate probabilistic models, which are challenging to obtain in practice. RL, on the other hand, struggles with extensive training requirements, poor generalization to new scenarios, and difficulties in handling hard constraints such as time windows and vehicle capacities, often necessitating complex reward engineering. A promising alternative is a hybrid approach that employs heuristics to address complex constraints and combinatorial high-dimensional problems while leveraging RL to create an adaptive framework for handling the problem's stochastic nature. This approach provides both generalization capabilities and efficiency in large-scale applications (Kallestad et al., 2023; Reijnen et al., 2024).

In this study, we extend previous research on the e-waste collection problem by introducing a new problem formulation that accounts for stochastic travel times, referred to as the Heterogeneous Vehicle Routing Problem with Multiple Time Windows and Stochastic Travel Times (HVRP-MTWSTT). This formulation incorporates various real-world constraints, including a heterogeneous fleet and multi-period planning with time windows. To solve the HVRP-MTWSTT efficiently for practical-sized instances, we propose a novel method combining Deep Reinforcement Learning (DRL) with local search-based heuristics, termed DRL-LSH. Compared to metaheuristic methods, DRL-LSH provides a more flexible and adaptive framework by leveraging information generated during the search process to enhance the search strategy. In comparison to hyperheuristics, DRL-LSH utilizes the powerful learning capabilities of DRL to improve the selection of appropriate heuristic operators. Furthermore, compared to previous hybrid approaches (Kallestad et al., 2023; Reijnen et al., 2024), DRL-LSH extends beyond enhancing the adaptive operator selection layer of ALNS to offer a more general solution framework that integrates diverse classes of problem-dependent heuristics and is applicable to real-world and complex VRP applications. We validate this approach by applying it to a real-world e-waste collection case study in Singapore and comparing its performance against other metaheuristic and hyperheuristic methods.

Our contributions are summarized as follows:

1. We introduce a stochastic formulation of the e-waste collection problem, referred to as the Heterogeneous Vehicle Routing Problem with Multiple Time Windows and Stochastic Travel Times (HVRP-MTWSTT). This formulation incorporates critical real-world constraints, including heterogeneous fleets, multiple time windows, multi-period planning, and stochastic travel times, reflecting the complexities of urban e-waste collection.
2. To address the challenges of solving complex and stochastic VRP variants like HVRP-MTWSTT, we propose a novel hybrid method, DRL-LSH, which integrates deep reinforcement learning with local search-based heuristics. DRL-LSH develops more effective search strategies in stochastic environments and demonstrates scalability to large-scale, real-world instances. We evaluated its performance through comparisons with multiple metaheuristic and hyperheuristic approaches and applied it to an e-waste collection case study in Singapore, showcasing its effectiveness and practicality in real-world applications.

The remainder of the paper is organized as follows. Section 2 provides background knowledge on both the target problem, i.e., HVRP-MTWSTT, and related algorithms, referencing the literature. Section 3 introduces the Mixed-Integer Linear Programming (MILP) model for HVRP-MTWSTT. The proposed algorithmic strategy, i.e., DRL-LSH, is discussed in detail in Section 4. In Section 5, we evaluate the proposed method and compare it with other reference algorithms using both synthetic and real-world instances. Finally, Section 6 concludes our study and discusses limitations and future research directions.

## 2. Related work

In this study, we position the relevance of this work within two key research lines. To the best of our knowledge, HVRP-MTWSTT represents the first formulation of the e-waste collection problem as a stochastic VRP variant. Developing a solution method for such a complex and stochastic VRP variant requires an approach capable of effectively handling intricate constraints and stochastic travel times while maintaining scalability for practical-sized instances. To address these challenges, we build on the concept of integrating DRL as an adaptive operator selection mechanism to dynamically choose among various local search-based heuristics. Based on this approach, we propose DRL-LSH — a hybrid method designed to solve the HVRP-MTWSTT problem efficiently. This method leverages the learning capabilities of DRL to ensure scalability for large-scale instances. We provide an overview of these research lines in the following sections.

### 2.1. Vehicle routing problem with the application in e-waste collection

The VRP is an important class of problems with many applications and has been extensively researched in the field of Operations Research for several decades. However, the application of VRP to e-waste collection has only recently gained attention due to the alarming increase in the amount of e-waste generated worldwide each year. The e-waste collection process presents unique challenges and features not encountered in other fields, necessitating more intensive research to develop efficient collection strategies. Belonging to the class of waste collection problems, e-waste collection shares characteristics such as

multiple periodic planning and the use of heterogeneous vehicles for collection tasks. The planning of e-waste collection over a multi-period horizon is a natural consequence of the fact that not all e-bins (fixed collection points) are collected daily (Hess et al., 2024). With the advancement of IoT, e-bins can be considered "smart bins", equipped with sensors that transmit trash level information to a centralized information system. This allows operators to decide whether an e-bin should be collected the following day or in the coming days, depending on the routing plan. Additionally, the use of heterogeneous vehicles in e-waste collection is a practical response to the variation in e-waste sizes and the differing accessibility of local roads. Beyond these characteristics, the e-waste collection process often involves two types of collection sites: on-demand collection requests from households and companies, and e-bins (Pourhejazy et al., 2021). The former is typically associated with preferred time windows requested by customers — an aspect that is not usually considered in municipal waste collection processes (Szwarc et al., 2021). Ensuring that e-waste is collected within customer-preferred time windows is a critical constraint in the e-waste collection problem, as it significantly impacts both revenue and customer satisfaction. Furthermore, on-demand requests introduce additional challenges due to their dynamic and variable locations and timings, adding complexity to the problem. The following is an overview of how previous research in the literature has addressed these aspects in the context of e-waste collection.

In previous research, modeling the e-waste collection process typically revolves around two variants of VRPs: VRP with Time Windows (VRPTW) and the Team Orienteering Problem (TOP) (Szwarc et al., 2021). When modeling e-waste collection as VRPTW, four algorithms were proposed in Nowakowski et al. (2018): Simulated Annealing, Tabu Search, Greedy, and improved Bee Colony Optimization. It concerns a collection support system and is tested using real-world data from Tokyo, Philadelphia, and Warsaw. For the integrated collection scheme that simultaneously accommodates on-call and door-to-door demands, an original capacitated general routing with time-window (CGRPTW) model was introduced in Pourhejazy et al. (2021). The mathematical model is formulated as Mixed-Integer Linear Programming (MILP), and Tabu Search is proposed to solve the problem. A model for e-waste collection encompassing both households and mobile collection was proposed in Król et al. (2016). This problem was addressed as a VRP with intermediate facilities, heterogeneous vehicles, considerations for breaks between shifts, and regional cost differences. Genetic algorithms were utilized, with a case study from Poland. Recently, the more complex e-waste collection problem was investigated (Gunawan, Nguyen, Nguyen et al., 2023). The problem is formulated as a heterogeneous VRP, accounting for multiple collection periods with varying time windows. This was termed the Heterogeneous VRP with Multiple Time Windows (HVRP-MTW). The collection points consider both customer requests and public drop-off e-waste collection points (e-bins). A hybrid metaheuristic, based on a Greedy Randomized Adaptive Search Procedure reinforced by Path-Relinking (GRASP-PR), was proposed to solve newly developed instances. It outperformed CPLEX in terms of both solution quality and computational time.

While previous studies have modeled the e-waste collection problem as a VRP and incorporated complex real-world constraints, a significant gap in real-world e-waste collection systems is the lack of consideration for uncertainty in travel time. This oversight significantly impacts service reliability, operational efficiency, and cost-effectiveness. The importance of incorporating stochastic travel time arises from the inherent characteristics of the e-waste collection problem — multi-period planning, a heterogeneous fleet of vehicles, time-dependent tasks, and the strategic placement of e-bins — all of which influence vehicle travel times, arrival schedules, and, ultimately, the overall objective value of the solution. In contrast, stochastic VRPs have been extensively studied in other domains. For comprehensive reviews of stochastic (and dynamic) VRPs, we refer readers to the surveys by Ritzinger et al. (2016) and Soeffker et al. (2022). Modeling the e-waste collection problem as a variant of the stochastic VRP, while retaining its specific and complex constraints, is expected to advance more realistic and practical approaches in this field of research.

### 2.2. Adaptive operation selection for VRP

Recently, the rapid growth of the Machine Learning (ML) field has created significant opportunities to enhance the performance of existing optimization algorithms or solve combinatorial optimization problems using end-to-end ML approaches. To keep our literature review concise, we will focus specifically on our method of using Reinforcement Learning to enhance heuristic-based methods for solving the VRP. For a broader overview of the field, we refer readers to the surveys by Bengio et al. (2021), Hildebrandt et al. (2023), and Karimi-Mamaghan et al. (2022). Other approaches beyond adaptive operator selection for solving the stochastic and dynamic VRP can be found in Baty et al. (2024), Serrano et al. (2024) and Zhang, Luo et al. (2023). For other ML-based approaches to stochastic and dynamic VRP, readers can refer to the competition held at IJCAI 2021, as discussed in Zhang, Bliek et al. (2023).

Before delving into the related works of our proposed method, it is worth to distinguish between low-level heuristics, metaheuristics, and hyper-heuristics, which are different types of methods applied to search for solutions to combinatorial optimization problems. Low-level heuristics refer to problem-specific techniques designed for particular problems (Pillay & Qu, 2018). Meanwhile, metaheuristics are a class of approximate optimization methods that orchestrate and guide the interaction between local improvement techniques and higher-level strategies, creating an iterative search process capable of escaping local optima and conducting a robust exploration of the search space (Gendreau et al., 2010). Finally, hyper-heuristics govern the selection of low-level heuristics based on the region of the solution space being explored (Martí et al., 2024). While metaheuristics require expertise to be tailored for specific problems, hyper-heuristics provide generalized solutions across problem domains by working on the heuristic space instead of the solution space (Ozcan et al., 2009). Their problem-independent nature is maintained through a domain barrier, which restricts explicit knowledge of the problem being solved. Fig. 1 illustrates the differences in generality between metaheuristics and hyper-heuristics.

In hyper-heuristics, the step of selecting heuristics largely depends on the concept of Operator Selection (OS), which is motivated by the fact that the efficiency of individual operators can vary across different regions of the search space (Drake et al., 2020; Fialho, 2010). An operator may perform well in certain search regions but poorly in others. Therefore, selecting the most appropriate operator at the right time can enhance performance by balancing exploration and exploitation. Metaheuristics have also employed OS strategies, as exemplified by Adaptive Large Neighborhood Search (ALNS) (Ropke & Pisinger, 2006), which builds on the Large Neighborhood Search (LNS) metaheuristic (Shaw, 1998) through the use of OS. Beyond the fundamental concept of OS, Adaptive Operator Selection (AOS) (Fialho, 2010) dynamically chooses operators during the search process. From an algorithmic perspective, AOS can be considered a branch of Automated Algorithm Design. For further exploration in this area, we recommend (Adriaensen et al., 2022; Di Liberto et al., 2016).

In connection with the AOS' exploration and exploitation focus, Reinforcement Learning (RL) has been the leading ML approach to perform AOS. The RL algorithms are designed to learn (near-) optimal policies by iteratively interacting with an environment. Various techniques have been devised to effectively manage the exploration–exploitation trade-off. Concerning AOS, RL primarily serves as a Credit Assignment (CA) mechanism (Gunawan et al., 2018; Lu et al., 2020). It has been especially effective, both in terms of solution quality and convergence speed, on the problem instances of increasing size (dos Santos et al., 2014; Zhao et al., 2021).

Regarding AOS in metaheuristics, ALNS is one metaheuristic that incorporates an adaptive layer into the search procedure, recording the past performance of pairs of "removal" and "insertion" operators for future selection. However, the adaptive layer of ALNS typically only has

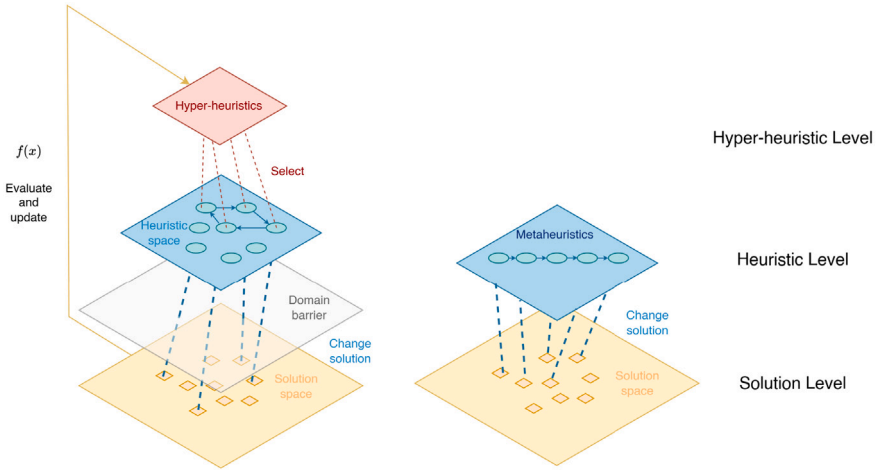



**Fig. 1.** Metaheuristics and hyper-heuristics in different generality levels.

a minor impact on the solution quality (Hemmati & Hvattum, 2017). Furthermore, the information provided for the adaptive layer is derived solely from the past performance of heuristics based on the generated solution quality. This can limit its decision-making capability due to a lack of comprehensive information about the current state of the search. ALNS also faces challenges when selecting operators from a large pool of problem-dependent heuristics. Consequently, recent studies have employed DRL as a selection mechanism for the ALNS framework. For instance, a DRL Hyperheuristic (DRLH) (Kallestad et al., 2023) was compared with ALNS and Uniform Random Selection (URS) on three routing problems: the Capacitated VRP (CVRP), the Pickup and Delivery Problem (PDP), and the PDP with Time Windows (PDPTW). DRLH outperformed ALNS in selecting operators from a large pool in fewer iterations for all those problems. Moreover, the performance gap between DRLH and the two reference algorithms widened as the problem size increased, making DRLH particularly suitable for real-world scenarios. Another study (Reijnen et al., 2024) employed DRL in ALNS for the Time-dependent Orienteering Problem with Stochastic Weights and Time Windows (TD-OPSWTW). DRL-ALNS outperformed two other ALNS variants in terms of solution quality and speed in comparison to the end-to-end DRL approaches. In both research efforts that integrated DRL into ALNS, the Proximal Policy Optimization (PPO) algorithm (Schulman et al., 2017) was utilized.

Our motivation for using a DRL-based AOS approach lies in the limitations of existing methods for stochastic VRPs. Multi-stage stochastic optimization methods are computationally expensive for large-scale applications, while end-to-end RL approaches struggle with high-dimensional combinatorial problems (Baty et al., 2024). A hybrid method that combines DRL with optimization methods is a promising alternative. Recent studies have explored the integration of DRL with ALNS to enhance its adaptive layer, focusing on macro-level operator control and improving generalization. While these approaches are well-suited for generic VRPs, they may not fully capture application-specific challenges. In this work, we address HVRP-MTWSTT, a real-world VRP variant that incorporates complex constraints and stochastic travel times. Solving this problem efficiently at scale requires fine-grained, micro-level route adjustments to meet its intricate constraints. Local search (LS), a key component in successful metaheuristics like GRASP and Iterated Local Search, offers efficient, localized optimization at the solution level. By combining DRL with LS, we propose a powerful, adaptive framework tailored for HVRP-MTWSTT, enabling scalability and effective handling of stochastic and large-scale applications.

## 3. Problem definition

The Heterogeneous Vehicle Routing Problem with Multiple Time Windows and Stochastic Travel Times (HVRP-MTWSTT) can be represented as a weighted and complete directed graph $G = (N, A)$ in which $N = \{0, 1, \ldots, n, n+1\}$ is the set of nodes and $A = \{(i, j) : i, j \in N, i \neq j, i \neq n+1, j \neq 0\}$ is the set of arcs. In this graph, two nodes 0 and $n+1$ represent the departing and returning depots, which can be identical nodes. The e-waste collection processes are conducted in $|K|$ days with $K = \{1, 2, \ldots, |K|\}$ is the set of collection days.

The problem involves two types of collection locations: on-demand collection requests from households or customers' doorsteps $N^1 \subset N$ and public drop-off locations (e-bins) $N^2 \subset N$. The e-bins are a type of smart bins equipped with sensors that can send information to the system about the amount of e-waste collected. Each day, the system updates the information on which e-bins need to be emptied, enabling staff to identify the bins requiring collection and the quantity of e-waste in each. Generally, the amount of e-waste usually does not exceed the e-bin capacity, so we do not consider the e-bin capacity in our model. Additionally, on-demand collection requests are made through a mobile application, where customers specify the amount of e-waste they have and the time of day they are available for collection. Therefore, the e-bins are available throughout the entire $|K|$ days, while customer requests are available only within a time window $[l_i^k, u_i^k]$ on each day $k$, where $l_i^k$ and $u_i^k$ represent the lower and upper bounds of the time window for customer $i$ on day $k$. For on-demand collection services, each customer $i$ will be charged an amount of $p_i$ for the individual request. Each collection location has a demand $d_i$ which requires the amount of time of $s_i$ to serve. In practice, due to diverse operational and logistical requirements such as e-waste characteristics, accessibility, environmental considerations, and regulatory compliance, e-waste collection tasks are typically carried out using a heterogeneous fleet of vehicles. In this model, this setting is simplified by representing the fleet as a set of heterogeneous vehicles $V = \{1, 2, \ldots, |V|\}$ employed for the collection task. Each vehicle $v \in V$ is characterized by a capacity $q^v$ and an operational cost $c^v$ per unit of time. To simplify the modeling process, we do not include constraints related to road width and regulatory compliance in our model.

Due to the unique characteristics of the e-waste collection problem, such as multi-period collection, the use of heterogeneous vehicles, the strategic placement of e-bins in crowded areas, and operations conducted in urban settings, the travel time of vehicles on each edge is modeled as a stochastic variable. Let $\tilde{t}_{ij}$ represent the stochastic travel time associated with arc $(i, j)$. These stochastic travel times are only revealed after the vehicles traverse the respective arcs. Since there is no prior knowledge about the distribution of travel times on the road network, we adopt the approach proposed by Reijnen et al. (2024) to model the stochastic travel times. Specifically, $\tilde{t}_{ij}$ is computed by multiplying the Euclidean distance $d_{ij}$ by a noise term $\eta$, which follows a discrete uniform distribution $\mathcal{U}\{1, 100\}$, and normalizing the result by

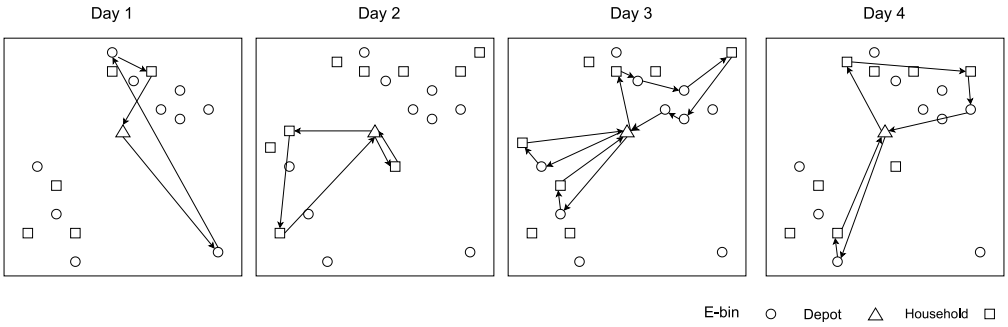



**Fig. 2.** A solution of the HVRP-MTWSTT with four collection days.

a scaling factor $\beta = 100$. The stochastic travel times $\tilde{t}_{ij}$ are sampled as follows:

$$\tilde{t}_{ij} = d_{ij}\frac{\eta}{\beta} \tag{1}$$

Due to the stochastic travel time, the arrival time $\tilde{a}_i$ at node $i$ is also a random variable. Therefore, when a vehicle arrives at customer $i$ before the lower bound of its time windows $l_i^k$, the vehicle $v$ has to wait for serving customer $i$, and an idle fee $\psi^v$ per unit of time will be imposed. On the other hand, if a vehicle arrives at customer $i$ after upper bound $u_i^k$, a penalty fee $\phi^v$ per unit of time will be imposed. The arrival time at node $j$ when traveling from customer $i$ is calculated by:

$$\tilde{a}_j = \max\{\tilde{a}_i, l_i^k\} + s_i + \tilde{t}_{ij} \tag{2}$$

Fig. 2 illustrates a possible HVRP-MTWSTT solution. The mathematical model is formulated below with the variables listed and summarized in the supplementary materials (Part A).

$$\max \sum_{i\in N^1} p_i - \mathbb{E}\left\{ \sum_{i\in N\setminus\{n+1\}, j\in N\setminus\{0\}, k\in K, v\in V} \left[(\tilde{t}_{ij}+s_j)c^v + \tilde{\delta}_j^{kv}\psi^v + \tilde{\zeta}_j^{kv}\phi^v\right] x_{ij}^{kv} \right\} \tag{3}$$

$$s.t. \quad \sum_{k\in K}\sum_{v\in V}\sum_{i\in N\setminus\{n+1\}} x_{ij}^{kv} = 1 \quad \forall j \in N \tag{4}$$

$$\sum_{v\in V}\sum_{i\in N\setminus\{n+1\}} x_{ij}^{kv} \le y_j^k \quad \forall j \in N,\ k \in K \tag{5}$$

$$\sum_{k\in K}\sum_{j\in N\setminus\{n+1\}} x_{0,j}^{kv} = 1 \quad \forall v \in V \tag{6}$$

$$\sum_{k\in K}\sum_{i\in N\setminus\{n+1\}} x_{i,n+1}^{kv} = 1 \quad \forall v \in V \tag{7}$$

$$\sum_{j\in N\setminus\{0\}} x_{0,j}^{kv} \le 1 \quad \forall k \in K,\ v \in V \tag{8}$$

$$\sum_{i\in N\setminus\{n+1\}} x_{i,n+1}^{kv} \le 1 \quad \forall k \in K,\ v \in V \tag{9}$$

$$\sum_{v\in V}\sum_{j\in N\setminus\{0\}} x_{0,j}^{kv} - \sum_{v\in V}\sum_{i\in N\setminus\{n+1\}} x_{i,n+1}^{kv} = 0 \quad \forall k \in K \tag{10}$$

$$\sum_{k\in K}\sum_{i\in N\setminus\{n+1\}} x_{ij}^{kv} - \sum_{k\in K}\sum_{i\in N\setminus\{0\}} x_{ji}^{kv} = 0 \quad \forall j \in N\setminus\{0, n+1\}, \forall v \in V \tag{11}$$

$$\sum_{i\in N\setminus\{n+1\}} x_{ij}^{kv} - \sum_{i\in N\setminus\{0\}} x_{ji}^{kv} = 0 \quad \forall j \in N, k \in K, v \in V \tag{12}$$

$$L_j^{kv} - L_i^{kv} - q^v\left(1 - x_{ij}^{kv}\right) \le d_j \quad \forall i \in N\setminus\{n+1\}, j \in N\setminus\{0\}, k \in K, v \in V \tag{13}$$

$$L_j^{kv} - L_i^{kv} + q^v\left(1 - x_{ij}^{kv}\right) \ge d_j \quad \forall i \in N\setminus\{n+1\}, j \in N\setminus\{0\}, k \in K, v \in V \tag{14}$$

$$L_j^{kv} \le q^v \sum_{i\in N\setminus\{n+1\}} x_{ij}^{kv} \quad \forall j \in N\setminus\{0\}, k \in K, v \in V \tag{15}$$

$$L_0^{kv} = 0 \quad \forall k \in K, v \in V \tag{16}$$

$$L_{n+1}^{kv} \le q^v \sum_{j\in N\setminus\{0\}} x_{0j}^{kv} \quad \forall k \in K, v \in V \tag{17}$$

$$\tilde{a}_j^{kv} - \tilde{a}_i^{kv} - s_i - M\left(1 - x_{ij}^{kv}\right) \le \tilde{t}_{ij} \quad \forall j \in N\setminus\{0, n+1\}, k \in K, v \in V \tag{18}$$

$$\tilde{a}_j^{kv} - \tilde{a}_i^{kv} - s_i + M\left(1 - x_{ij}^{kv}\right) \ge \tilde{t}_{ij} \quad \forall j \in N\setminus\{0, n+1\}, k \in K, v \in V \tag{19}$$

$$\tilde{a}_i^{kv} + \tilde{t}_{i,n+1} + \left(u_{n+1}^k - l_0^k\right)\left(1 - x_{i,n+1}^{kv}\right) \ge l_{n+1}^k \quad \forall i \in N\setminus\{n+1\}, k \in K, v \in V \tag{20}$$

$$\tilde{a}_i^{kv} + \tilde{t}_{i,n+1} \le u_{n+1}^k + \left(u_{n+1}^k - l_0^k\right)\left(1 - x_{i,n+1}^{kv}\right) \quad \forall i \in N\setminus\{n+1\}, k \in K, v \in V \tag{21}$$

$$\tilde{\delta}_i^{kv} = \begin{cases} l_i^k - \tilde{a}_i^{kv}, & \text{if } l_i^k - \tilde{a}_i^{kv} > 0\ \forall i \in N^1, k \in K, v \in V \\ 0, & \text{otherwise} \end{cases} \tag{22}$$

$$\tilde{\zeta}_i^{kv} = \begin{cases} \tilde{a}_i^{kv} - u_i^k, & \text{if } \tilde{a}_i^{kv} - u_i^k > 0\ \forall i \in N^1, k \in K, v \in V \\ 0, & \text{otherwise} \end{cases} \tag{23}$$

$$x_{ij}^{kv}, y_j^k \ \text{ binary } \forall i \in N\setminus\{n+1\}, j \in N\setminus\{0\}, k \in K, v \in V \tag{24}$$

$$L_i^{kv}, \tilde{a}_i^{kv}, \tilde{\delta}_i^{kv}, \tilde{\zeta}_i^{kv} \quad \text{non-negative} \quad \forall i \in N, v \in V \tag{25}$$

The objective function (3) aims to maximize the profit generated by the e-waste collection process, taking into account the income from on-demand collections as well as the costs associated with operations, idle time and penalties caused by uncertain travel times. Constraints (4) require that all e-bins and on-demand customers are visited by exactly one vehicle. Constraints (5) impose that on-demand customers can only be served during their available time periods. Constraints (6) to (9) specify that each vehicle departs and returns at most once during the specified collection period. Constraint (10) ensures that all the departed vehicles return to the depot on the same day. Constraints (11) and (12) imply that a vehicle arriving at node $j$ must leave it on the same day. Constraints (13) and (14) are conservation of flow constraints, which guarantee that they pick-up the exact amount of e-waste from node $i \in N\setminus\{0, n+1\}$. These constraints also eliminate sub-tours.

Constraints (15) show that the loads of each vehicle cannot exceed its maximum capacity at any node $i \in N\setminus\{0\}$. Constraints (16) initialize the load for each vehicle at depot 0. Constraints (17) shows that the load of vehicle $v \in V$ at depot $n+1$ must not exceed its capacity. Constraints (18) and (19) verify the travel time of the vehicles in the each time period $k \in K$. Constraints (20) and (21) ensure all vehicles return to the depot within the operation hour of Depot $n+1$. Constraints

(22) calculate the idle time $\tilde{\delta}_i^{kv}$ if vehicle $v \in V$ arrives earlier than the time window lower bound $l_i^k$ of node $i \in N^1$ on day $k \in K$. Constraints (23) calculate the penalty time if vehicle $v \in V$ arrives after the time window upper bound $u_i^k$ of node $i \in N^1$ on day $k \in K$. Constraints (24) and (25) are feasible regions of the decision variables.

The following part presents the linearization of two non-linear Constraints (22) and (23) by Constraints (26)–(30). Two binary decision variables $\kappa_i^{kv}$ and $\rho_i^{kv}$, indicating whether vehicle $v \in V$ arrives at node $i \in N^0$ earlier/later than the given time windows at day $k \in K$ or not, are introduced. Constraints (26) and (27) compute the idle time $\tilde{\delta}_i^{kv}$ for the early arrival of vehicle $v \in V$ at node $i \in N^1$ while Constraints (28) and (29) compute the penalty time $\tilde{\zeta}_i^{kv}$ for late arrival of vehicle $v \in V$ at node $i \in N^1$. Constraints (30) define the variable domains.

$$\tilde{\delta}_i^{kv} \leq M\kappa_i^{kv} \quad \forall i \in N, k \in K, v \in V \tag{26}$$

$$l_i^k - \tilde{a}_i^{kv} \leq \tilde{\delta}_i^{kv} \leq l_i^k - \tilde{a}_i^{vk} + M(1-\kappa_i^{kv})\ \forall i,k,v \in N,K,V \tag{27}$$

$$\tilde{\zeta}_i^{kv} \leq M\rho_i^{kv}\ \forall i \in N, k \in K, v \in V \tag{28}$$

$$\tilde{a}_i^{kv} - u_i^k \leq \tilde{\zeta}_i^{kv} \leq \tilde{a}_i^{vk} - u_i^k + M(1-\rho_i^{kv})\ \forall i,k,v \in N,K,V \tag{29}$$

$$\kappa_i^{kv}, \rho_i^{kv} \quad \text{binary } \forall i \in N \setminus \{0\}, k \in K, v \in V \tag{30}$$

## 4. Deep reinforcement learning to control local search operators (DRL-LSH)

In this section, we introduce solution methods for the HVRP-MTWSTT that utilize deep reinforcement learning (DRL) to control local search operators throughout the search process, denoted as DRL-LSH. Specifically, we model the search process for solving the HVRP-MTWSTT as a sequential decision-making problem. Within this framework, deep reinforcement learning agent interacts with the search procedure, guiding the selection of appropriate heuristic operators based on the current state of the search.

The proposed DRL-LSH framework is illustrated in Fig. 3. Specifically, at each decision step, the DRL agent relies on the state of the search process, which includes various features related to the solution found and the search iteration. Since DRL-LSH uses a policy-gradient method for learning, it outputs a probability distribution over actions, which corresponds to different heuristic operators. The agent then samples an action from a set of heuristic operators $(H_1, \ldots, H_n)$ to apply to the current solution, aiming to enhance it. The reward is determined based on the quality improvement of the solution achieved by the chosen action. Using this collective information, which includes the state, action, and reward, the DRL agent updates its weights to learn the optimal actions for a given search state.

### 4.1. Modeling the search procedure as a Markov decision process

The search process is modeled as a Markov Decision Process (MDP), wherein $S$ represents the set of states, $R$ represents the reward function, and $A$ represents the set of actions. The primary goal of the DRL agent is to acquire a policy $\pi$, which maps a state $s_t$ to an action $a_t$ at time step $t$, ultimately aiming to maximize the expected cumulative future rewards. The explicit details concerning the state space $S$, action space $A$, and reward function $R$ are elaborated below.

**State Space**: The DRL agent will take actions based on the current state $s_t$ of the search. We formulate the state $s$ as a one-dimensional vector comprising 8 features, as described in Table 1. The state provides the agent with information about the differences between the newly found solution and the current solution, as well as the number of iterations without restarting and improvement. Additionally, since we use path-relinking as an action of DRL-LSH, we need to maintain an elite solution pool $P$ during the search. Therefore, the state $s_t$ will also contain the information whether the new solution is good enough to be included in the solution pool $P$ for the path-relinking procedure. This helps the DRL agent to become aware of the current search state and take actions to maximize returns.

**Actions**: In the proposed framework of DRL-LSH, the actions consist of various operators that explore different neighborhoods of the current solution and generate new solutions to escape from the local optimum. We utilize 13 different actions, which are explained below. The details can be found in the supplementary materials (Part B).

<*Action 0*> The first action is to construct a feasible solution. We employ a construction heuristic that is based on the Clarke and Wright heuristic, which is augmented with the Realistic Opportunity Savings - $\gamma$ approach for heterogeneous fleet, as proposed by Golden et al. (1984). While the Clarke and Wright heuristic is effective for handling capacity constraints, it does not address collection period constraints. For this issue, a repair operator is incorporated to ensure solution feasibility under collection period constraints (Gunawan, Nguyen, Nguyen et al., 2023). Additionally, since travel time is stochastic, we use the maximum travel time when constructing new solutions using the Clarke and Wright heuristic. With these modifications, our construction heuristic is capable of producing feasible solutions.

<*Actions 1 to 11*> They represent local search operators utilizing the following neighborhoods ($N1$ to $N11$) as a local search procedure in the first-improvement strategy:

*N1. Intra-relocation:* N1 explores the neighborhood solution by removing customer $i$ from route $v_i$ and then reinserting $i$ at another position in the same route $v_i$.

*N2. Inter-relocation:* N2 removes customer $i$ from its current route $v_i$ and reinserts $i$ to a new route $v_j$ provided that there is at least one day in the available period of $i$ that coincides with the collection day of $v_j$. Additionally, the total capacity of $v_j$ after the reinsertion must not exceed the capacity of the largest vehicle used.

*N3. Intra-swap:* N3 selects node $i$ in route $v_i$ and swaps it with another node $j$ in the same route.

*N4. Inter-swap:* N4 selects node $i$ from route $v_i$ and another node $j$ from a different route $v_j$. The selected nodes are swapped provided that their available period are compatible with their target routes' $v_i$ and $v_j$ collection day. Furthermore, the total demand of the two routes $v_i$ and $v_j$ after swapping must not exceed the maximum capacity of the largest vehicle in the fleet.

*N5. 2-opt (intra-route):* a new solution is built from an initial solution $x$ by removing two consecutive arcs $(i, i+1)$ and $(j, j+1)$ from route $v_i$. Then, two other arcs $(i, j)$ and $(i+1, j+1)$ are added to reconnect the selected routes. As a result, the subpath excluding the depot between $i+1$ and $j$ is inverted.

*N6. 2-opt* (inter-route):* the principle of generating a new solution using the N6 move is similar to that of N5. However, the N6 selects and removes two arcs $(i, i+1)$ and $(j, j+1)$ from two distinct routes $v_i$ and $v_j$, respectively. Two new arcs are then added to reconnect the selected routes, and both the inverted and non-inverted subpath versions of the new routes are considered. The N6 selects the higher-profit neighborhood between the two versions of new solutions. The condition of this action is that the two selected routes $v_i$ and $v_j$ have the same collection day and the two new routes after reconnecting have to satisfy the capacity constraint.

*N7. or-opt (inter-route):* the solution is defined by relocating a subpath $(i+1, \ldots, j)$ from route $v_i$ to another route $v_j$. The conditions for applying the move are the same as those for the N6.

*N8. 3-opt (intra-route):* N8 deletes three arcs $(i, i+1)$, $(j, j+1)$, and $(l, l+1)$ in a route $v_i$. It then considers all ways to reconnect these subpaths, including both inverted and non-inverted versions. In

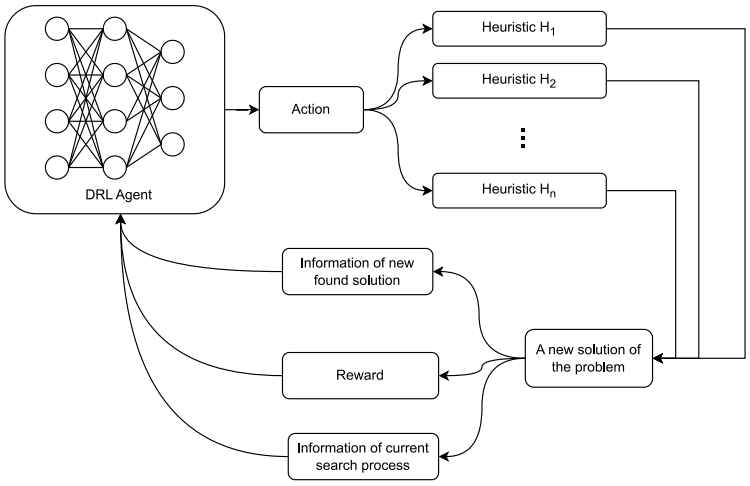



**Fig. 3.** DRL-LSH framework.

**Table 1**
State space features for DRL-LSH.

| Feature | Description |
|---|---|
| Previous action | Integer feature indicates the action taken in previous iteration |
| Local improvement | Binary feature indicates whether the current solution was accepted and better than previous solution |
| Global improvement | Binary feature indicates whether current found solution was better than the best solution so far |
| Current iteration | The current iteration of the search process |
| Acc_Restart | The number of consecutive iterations without restarting the search procedure |
| cost_diff_best | Percentage difference between the objective values of current and best solution |
| Acc_No_Impr | The number of consecutive iterations without improving the best-found solution |
| Update_to_P | Binary feature indicates whether current solution is allowed to enter elite solution pool or not |

total, eight neighborhoods are explored, and the action selects the one with the highest profit.

*N9. Add new route:* N9 randomly selects a node $i$ from route $v_i$ and removes it. Then, it creates a new single tour $v_i'$ from node $i$. The collection day $k_v$ for tour $v_i'$ is selected by comparing the profits of the solution if route $v_i'$ is carried out in different days within the collection period $K_v$. Next, the solution with the highest profit is selected. The condition for applying this action is still having unused vehicles.

*N10. Split to single:* N10 deletes two randomly selected nodes $i$ and $j$ from two distinct routes $v_i$ and $v_j$, respectively. It then creates a new tour by connecting those nodes. There are two neighborhoods created by N10 and the one with the higher profit is selected. The condition for this move is that the two selected nodes $i$ and $j$ must have at least one available day in common.

*N11. Combine tours:* N11 selects two random tours $v_i$ and $v_j$ that have a combined total capacity that is less than the maximum capacity of the largest vehicle in the fleet. It then combines these tours to create a new tour. Two neighborhoods are created by the N11, and the one with the higher profit is selected.

<*Action 12*> Path-relinking is an intensification mechanism (Glover, 1997). It relinks the current solution with an elite solution in the solution pool to find a promising solution.

**Reward Function**: At each time step, the environment sends the DRL agent a single number called a reward, which defines the goal of RL. Good reward strategies need to avoid the reward hacking behavior (Amodei et al., 2016) of the agent, where the agent keeps exploiting the reward function to increase the reward without actually optimizing the intended objective. Therefore, the reward function needs to balance the need for gradual and incremental rewards while avoiding the reward hacking behavior of the agent (Kallestad et al., 2023).

In the earlier studies integrating DRL into ALNS (Kallestad et al., 2023; Reijnen et al., 2024), the reward function utilizes the operator scoring system of the vanilla ALNS (Ropke & Pisinger, 2006). This system encourages the agent to find a new solution $x'$ better than the current solution $x$ and gives a higher reward if the new solution $x'$ is better than the best solution so far $x^*$. Additionally, a small reward is given to the agent if the newly found solution $x'$ is worse than the current solution $x$ but satisfies the acceptance criterion, e.g., SA acceptance condition. This reward function can work well in the ALNS environment but may lead to reward hacking in our framework. The DRL agent exploits the reward function by continuously creating a new solution (restarting the search) and applying only one improvement operator to improve this solution, then iterating these steps again to hack the reward. In our framework, we only encourage the DRL agent to explore new solutions $x'$ that are better than the best solution found so far $x^*$. The reward function $r^{EX}(s_t, s_{t+1})$ focuses on exploring new best solutions is defined as follows:

$$r^{EX}(s_t, s_{t+1}) = \begin{cases} \alpha^{EX}, & \text{if } f(x') > f(x^*) \\ 0, & \text{Otherwise.} \end{cases} \tag{31}$$

Here, $\alpha^{EX}$ is the score given to the DRL agent when the taken action explores a new solution $x'$, which is better than the best solution so far, $x^*$. In our study, the choice of $\alpha^{EX}$ is based on the reward function proposed in Kallestad et al. (2023) and Reijnen et al. (2024), where a score of 5 is given to the DRL agent when the action taken explores a new solution that is better than the best solution found so far. The $r^{EX}$ reward function operates similarly but is slightly modified compared to $R_t^{PM}$, which showed promising results (Kallestad et al., 2023) but exhibited instability when compared to the ALNS scoring system in their DRLH framework. However, the observations in Section 5 show the $r^{EX}$ reward function works stably and is suitable for our DRL-LSH framework. Moreover, we also consider another reward function, $r^{IM}$, which utilizes the objective value of the best solution achieved by a metaheuristic to assess and reward the DRL agent after it takes an action. This guides the DRL agent to improve the quality of the current solution by competing with the performance of the metaheuristic (see Section 5.3.5).

### 4.2. Training DRL-LSH

The training phase of DRL-LSH is presented in Algorithm 1. During training, for each step, the algorithm selects a problem instance and uses action 0 to generate a feasible solution denoted as $x$ (Lines 5–6). Simultaneously, the best solution $x^*$, the initial state $s_0$, and the solution pool $P$ — which will later store high-quality solutions during the path-relinking process — are initialized (Lines 7–9). At each time step, the Deep Reinforcement Learning (DRL) agent relies on the current state $s_t$ and policy $\pi_\theta$ to select action $a_t$ (Line 11). Subsequently, the algorithm applies the heuristic operator corresponding to the selected action $a_t$ on the current solution $x$, generating a new solution denoted as $x'$, and records the reward function value $r_t$ (Lines 12–13). Then, the newly generated solution is added to the solution pool $P$, if it satisfies the condition: $f(x') > (1-\mu)f(x^*)$, where $\mu$ represents the diversification coefficient (Lines 14–15).

**Algorithm 1** DRL-LSH - Training

1: **Input:** number of training steps, $M$;
2: Time horizon, $i_{max}$
3: $step = 0$;
4: **while** $step < M$ **do**
5: Initialize a problem instance;
6: Create a feasible solution, $x$;
7: Initialize best solution, $x^*, f(x^*) \leftarrow x, f(x)$;
8: Initialize a random initial state $s_0$
9: Initialize pool P for path-relinking
10: **for** $i = 0$ to $i_{max}$ **do**
11: $a_t \leftarrow$ action given by policy $\pi_\theta$ based on state $s_t$;
12: Take action $a_t$;
13: Observe reward $r_t$ and new solution $x'$;
14: **if** $f(x') > (1-\mu)f(x^*)$ **then**
15: Add $x'$ to $P$;
16: **end if**
17: **if** action $a_t$ create a new solution **then**
18: $x, f(x) \leftarrow x', f(x')$;
19: **end if**
20: **if** $f(x') > f(x^*)$ **then**
21: $x^*, f(x^*) \leftarrow x', f(x')$;
22: **end if**
23: $step = step + 1$
24: Add $(s_t, a_t, \pi_\theta(a_t|s_t), V_\phi(s_t), r_t)$ to memory $\mathcal{H}$
25: Update new state $s_{t+1}$
26: **end for**
27: Update policy $\pi_\theta$
28: **end while**

If the newly obtained solution $x'$ surpasses the current best solution $x^*$, it becomes the new best solution for the subsequent iterations (Lines 20–21). At the end of each time step, all information related to the current state $s_t$, the taken action $a_t$, the action probability distribution $\pi_\theta(a_t|s_t)$, and the value associated with the state $V_\phi(s_t)$ (where $\phi$ represents the parameters of the critic network) are stored in the memory buffer $\mathcal{H}$. Additionally, the new state $s_{t+1}$ is updated based on the progress of the search procedure (Lines 24–25). Every $i_{\max}$ iterations, the algorithm samples mini-batches from the memory buffer $\mathcal{H}$ to perform updates on the policy $\pi_\theta$ (Line 27).

### 4.3. Learning algorithm

In this study, we utilize Proximal Policy Optimization (PPO) (Schulman et al., 2017) for updating the weight $\theta$ of policy network $\pi_\theta$ of Algorithm 1. PPO is a widely applied and efficient policy gradient algorithm employed in both continuous and discrete environments (Kallestad et al., 2023; Reijnen et al., 2024). PPO focuses on maximizing the policy improvement while mitigating the risk of performance collapse through controlled policy updates using a "clipped" surrogate objective that penalizes changes in the policy when the new policy deviates too far from the old one, based on a clipping parameter. The implementation of the Proximal Policy Optimization (PPO) algorithm is outlined in the supplementary materials (Part C).

The input to the DRL-LSH algorithm is the state $s$ and its output is a probability distribution over actions. Due to the adoption of a multi-discrete action space in our DRL-LSH algorithm, the output layers are fully connected and employ a softmax activation function. This function generates a probability distribution over available actions as opposed to a single output, effectively determining the policy distribution. PPO is a type of actor–critic algorithm that utilizes two neural networks: a policy network and a critic network. The policy network acts as the actor that proposes a set of possible actions given a state. The critic network estimates the value function, which evaluates the action taken by the actor based on the given policy. The loss function of the policy network, with parameters $\theta$, in the PPO objective function is defined below:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(\rho_t(\theta)\hat{A}_t,\ clip(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right] \tag{32}$$

where $\rho_t(\theta)$ is the probability ratio of taking actions under the new policy compared to the old policy and $\hat{A}_t$ is an estimator of the advantage function at timestep $t$:

$$\rho_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{33}$$

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1} \tag{34}$$

where $\delta_t = r_t + \gamma V(s_{t+1} - V(s_t))$ with $r_t$ is reward and $V(s_t)$ is the value at state $s_t$ and $t$ specifies the time index in $[0, T]$, within a given length-$T$ trajectory segment. The second term in the loss function, $clip(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t$, modifies the surrogate objective by clipping the probability ratio. The $\epsilon$ is a hyperparameter. This adjustment eliminates the incentive for pushing $\rho_t$ beyond the interval $[1-\epsilon, 1+\epsilon]$. The clipped objective is subsequently compared with the unclipped objective to choose the lesser value. This strategy ensures that overly substantial policy updates are avoided during the agent's training process. $\gamma$ is the discount factor while $\lambda$ is a smoothing parameter for reducing the variance in training, offering stability. To account for multiple discrete actions, the policy loss is calculated separately for each action and summed to obtain a single scalar loss value. The clipping function is applied independently for each action. Furthermore, an entropy bonus, denoted as $E[\pi_\theta(s_t)]$, with a coefficient of $c_E$, is incorporated into the objective function of the policy network to encourage adequate exploration.

The critic loss of critic network, with parameters $\boldsymbol{\Phi}$, is given by the squared error between the predicted value $V_\phi(s_t)$ and the actual discounted sum of rewards obtained from the current state $\hat{R}_t$:

$$L^{VF}(\phi) = \hat{\mathbb{E}}_t \left[(V_\phi(s_t) - \hat{R}_t)^2\right] \tag{35}$$

## 5. Computational experiments

In this section, we provide a comprehensive computational experiment designed to assess the proposed solution method. We delve into the benchmark instances, reference algorithms, experiment settings, and present the experimental results. Furthermore, we conduct a comparative analysis to benchmark the performance of DRL-LSH in relation to hyperheuristic and metaheuristic methods. Additionally, the convergence, scalability, and learned policies of DRL-LSH will be further analyzed.

### 5.1. Benchmark instances

The proposed HVRP-MTWSTT builds upon the deterministic problem introduced by Gunawan, Nguyen , Yu et al. (2023). Consequently, the benchmark instances have also been derived from this deterministic version. These instances were originally adapted from the benchmark instance sets created by Homberger and Gehring (1999). An instance refers to a specific configuration of the problem, characterized by a defined set of locations, demands, vehicle capacities, costs, income, collection periods, service times, and time windows. From the benchmark

**Table 2**
Summary of test datasets used in experiment.

| Datasets | $N_1$ | $N_2$ | $V_1$ | $V_2$ | No. of instances |
|---|---|---|---|---|---|
| Instance20 | 10 | 10 | 5 | 5 | 200 |
| Instance50 | 25 | 25 | 13 | 13 | 200 |
| Instance100 | 50 | 50 | 25 | 25 | 200 |

**Table 3**
Summary characteristic of instances for the scalability experiment.

| No | Instance | $N_1$ | $N_2$ | $V_1$ | $V_2$ |
|---|---|---|---|---|---|
| 1 | S–8 | 4 | 4 | 4 | 4 |
| 2 | S–12 | 6 | 6 | 3 | 3 |
| 3 | S–16 | 8 | 8 | 4 | 4 |
| 4 | S–20 | 10 | 10 | 5 | 5 |
| 5 | S–24 | 12 | 12 | 6 | 6 |
| 6 | S–30 | 15 | 15 | 8 | 8 |
| 7 | S–40 | 20 | 20 | 10 | 10 |
| 8 | S–50 | 25 | 25 | 13 | 13 |
| 9 | S–60 | 30 | 30 | 15 | 15 |
| 10 | S–70 | 35 | 35 | 18 | 18 |
| 11 | S–80 | 40 | 40 | 20 | 20 |
| 12 | S–90 | 45 | 45 | 23 | 23 |
| 13 | S–100 | 50 | 50 | 25 | 25 |
| 14 | S–110 | 55 | 55 | 28 | 28 |

instances of the deterministic problem, we selected three instances, each representing different problem sizes with 20, 50, and 100 nodes. These are referred to as "selected instances". For each selected instance, we generated 400 "synthetic instances" by introducing randomized variations in e-bins, customer coordination points, and time windows. 50% of these 400 synthetic instances were used for training, while the remaining 50% were reserved for testing the model. The training and testing ratio were chosen based on the methodology described by Reijnen et al. (2024). The introduction of randomness into the stochastic travel times occurs during the computational process of the problem instance. The main characteristics of test instances are summarized in Table 2. $N_1$ and $N_2$ denote the number of customers and e-bins in the graph, while $V_1$ and $V_2$ represent the number of vehicle types I and II in the fleet, respectively. The specifications corresponding to vehicle type I include a capacity $q^1$ of 500, an operating cost $c^1$ of 2, a penalty cost $\phi^1$ of 7, and an idle cost $\psi^1$ of 3, whereas for vehicle type II, the corresponding values are 100, 1, 2, and 5, respectively. We consider a 4-day collection period ($|K|$) in all benchmark instances. The instances will be provided upon request.

For the scalability experiment (Section 5.3.3), we designed 14 additional instances, each featuring variations in the number of nodes, vehicles, e-bins, customer coordination points, and time windows. The key characteristics of the instances are outlined in Table 3. The remaining parameters, such as operating cost, penalty cost, idle cost of vehicles, and the horizon length of the collection period, remain consistent with those of the instances in the training dataset. Further details on the experimental settings are presented in the supplementary materials (Part D).

### 5.2. Reference algorithms

We compared the proposed approach against metaheuristic and hyperheuristic algorithms (called as "reference algorithms"), which are established methods commonly used as benchmarks to evaluate new approaches. These algorithms serve as a baseline for assessing the performance of the proposed method, particularly in terms of solution quality and computational efficiency. Specifically, we implemented five reference algorithms to solve the HVRP-MTWSTT: three metaheuristic methods and two hyperheuristic methods. These methods serve as frameworks for managing the low-level heuristics. In the context of RL, the heuristics are considered actions to be selected and performed within the DRL-LSH framework. We chose not to use end-to-end DRL approaches as reference algorithms, as many of these methods face challenges with high-dimensional combinatorial problems, have been tested only on relatively small scales, or are not adaptable to our specific problem. Consequently, they are not applicable to our experiments and do not offer a suitable evaluation for DRL-LSH.

The first reference algorithm is the Greedy Randomized Adaptive Search Procedure (GRASP) complemented with path-relinking (GRASP-PR), as proposed in Gunawan, Nguyen, Nguyen et al. (2023) for solving the deterministic version of HVRP-MTWSTT. While the original algorithm is not designed for stochastic problems, we organize the actions used for DRL-LSH in a similar manner to GRASP-PR for the purpose of comparison. The second reference algorithm is GRASP tuned by Q-learning (Nguyen et al., 2024), where the state–action value is used to determine the order of local search operators within the GRASP framework. The third reference algorithm is the integration of GRASP with Monte Carlo simulation (SimGRASP) (Festa et al., 2018), designed to solve VRP with stochastic demand using Monte Carlo simulation to evaluate the local optimum found by GRASP. The fourth reference algorithm is the Look-ahead hyper-heuristic (HH-probe) (Meignan et al., 2016), which employs a pre-selection phase (probe) to assess the potential costs that different local search operators can achieve within the same probe time. The final reference algorithm employed is the Q-learning based hyper-heuristic (HH-Q) (Nguyen et al., 2024), which leverages information from a trained Q-table to formulate the stochastic policy with biased randomized action selection to choose a low-level heuristic. As SimGRASP is designed to address the VRP with stochastic demand, and HH-Probe is evaluated on the classic VRP, we adapt these concepts to solve the HVRP-MTWSTT using the operators that serve as actions for DRL-LSH.

### 5.3. Experimental results

Due to the stochastic nature of DRL-LSH, we conducted training by using four different random seeds. In this process, we employed three separate algorithms: DRL-LSH20, DRL-LSH50, and DRL-LSH100, each trained on a set with different instance sizes: 20, 50, and 100. Each algorithm underwent training with a diverse set of 200 problem instances. Across all three instance groups, the DRL agent demonstrates convergence after 2.4 million learning steps. In terms of training time, DRL-LSH20, DRL-LSH50, and DRL-LSH100 took 8.47 h, 12.7 h, and 15 h, respectively.

Fig. 4 illustrates the rolling mean of training rewards over these 2.4 million steps, with varying seeds. The reward at each training step indicates the algorithm's ability to explore new best solutions. Therefore, a higher reward indicates a greater number of times where the algorithm explores new best solutions. A stable rolling mean reward at a high level indicates that the trained policy has the capability to explore more new best solutions (improved intensification) compared to the untrained policy. As observed in the figure, the convergence patterns of the DRL agent remained consistent among the three instance groups. In the initial stage, DRL agents experience rapid improvements before stabilizing at a high level of reward. It can be seen that the algorithms trained on larger instances have better rewards per episode than those trained on smaller instances. This can be attributed to the fact that larger instances have considerably larger solution spaces than the smaller ones. Therefore, within 6000 actions, DRL-LSH20 reaches its limit in the number of times it can explore new best solutions.

Additionally, the rewards per episode for DRL-LSH50 and DRL-LSH100 happen to be close. With 6000 search actions in the larger solution spaces of instance50 and instance100, DRL-LSH50 and DRL-LSH100 can still achieve higher rewards. However, since the number of search actions is capped at 6000, their average rewards converge to similar levels. We predict that the gap in rewards between DRL-LSH50 and DRL-LSH100 will be larger if more actions are performed. Ultimately, all three models converged around a rolling mean reward range from 95 to 115.

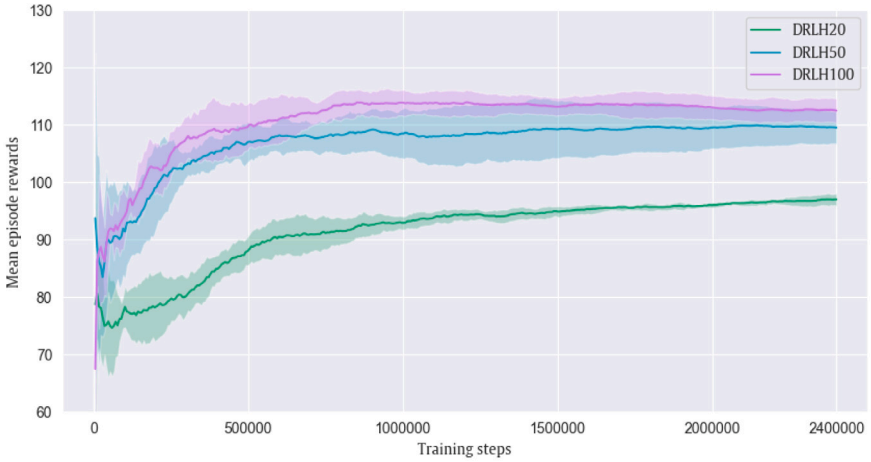



**Fig. 4.** Training reward over 2.4 million steps on three groups of instances.

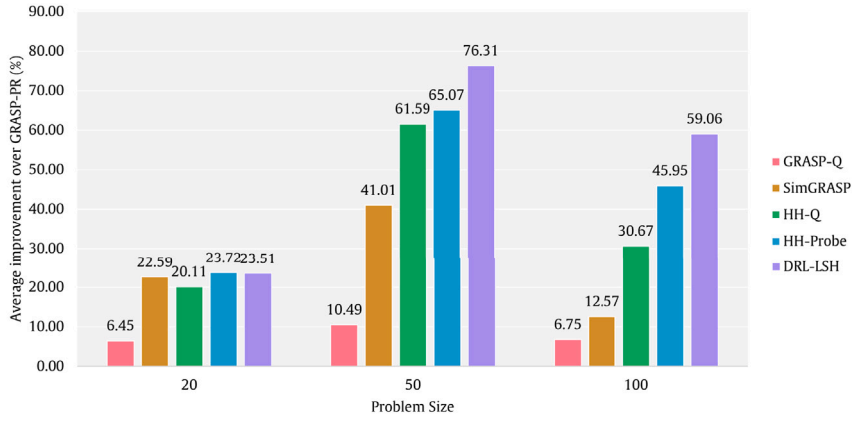



**Fig. 5.** Performance of DRL-LSH in solving 600 different instances.

The trained algorithm undergoes a comprehensive evaluation through a series of experiments, assessing performance and scalability in comparison to various reference algorithms on the HVRP-MTWSTT. Initially, we deployed the trained algorithms to address three groups of test instances with problem sizes of 20, 50, and 100, and subsequently, the performance of DRL-LSH is compared with the performance of the reference algorithms in Section 5.3.1. In Section 5.3.2, we focus on presenting information regarding the convergence speed of DRL-LSH in comparison with other reference algorithms. To evaluate the ability of DRL-LSH in solving previously unseen instances with additional nodes and vehicles, we conduct a scalability experiment wherein DRL-LSH is employed to address 14 different instances with sizes ranging from 8 to 110 nodes (Section 5.3.3). Section 5.3.4 provides detailed insights into the learned policies and state–action t-SNE visualization of DRL-LSH. We also present an experiment involving another reward function in Section 5.3.5, alongside with a statistical test based on non-parametric tests and post-hoc analysis.

### 5.3.1. Experiment on test instances

Each method was evaluated on three sets of instances with problem sizes of 20, 50, and 100 nodes, respectively, each set containing 200 instances. Table 4 indicates the performance of different methods across three instance sets, including the minimum profit (Min), maximum profit (Max), average profit (Avg. profit), standard deviation (Std), the number of instances where the method achieves the best solution (Nr. Best), and computational time (CPU) when solving 200 problem instances in each instance set. The performance gaps of GRASP-Q, SimGRASP, HH-Q, HH-Probe, and DRL-LSH in relation to GRASP-PR when solving 600 different problem instances are indicated in Fig. 5.

For small-sized instances (Instance20), the performances of SimGRASP, HH-Q, HH-Probe, and DRL-LSH are relatively similar, with all these methods achieving approximately a 20% improvement in average profit over GRASP-PR. The competitive performance of these methods at this scale suggests that AOS methods, such as HH-Q, HH-Probe, and DRL-LSH, do not yet fully demonstrate their advantage over traditional metaheuristic methods in smaller problem spaces. Among the traditional metaheuristics, GRASP-Q shows a modest improvement of 6%–10% over GRASP-PR due to its adaptive ordering of operators, but it still lags behind the AOS-based methods. SimGRASP, originally designed for stochastic VRP, performs fairly well in small-sized instances due to its sample-based evaluation strategy. However, as the solution space expands, the limitations of its sample-based approach become evident.

As the problem size increases to medium-sized instances (Instance50), a clear performance gap emerges between DRL-LSH and other methods. DRL-LSH outperforms the other approaches, achieving the highest average profit of −512.287, representing an 11.24% improvement over HH-Probe, the second-best performing method. The widening performance gap can be attributed to the ability of DRL-LSH to dynamically select appropriate low-level heuristics based on the state of the problem, unlike traditional metaheuristic methods, which rely on a fixed sequence of operator applications. The results indicate that GRASP-PR and GRASP-Q, both of which apply a rigid sequence of operators, are less effective in handling the increased complexity of medium-sized instances, resulting in significantly lower performance compared to AOS methods.

**Table 4**
Performance comparison of different methods for solving 600 instances of varying sizes.

| Instance sets | Metric | GRASP-PR | GRASP-Q | SimGRASP | HH-Q | HH-Probe | DRL-LSH |
|---|---|---|---|---|---|---|---|
| Instance20 | Min | −93.81 | −69.57 | 145.46 | −156.85 | 80.49 | −36.44 |
| | Max | 1648.96 | 1620.94 | 1664.11 | 1611.74 | 1897.21 | 1825.1 |
| | Avg. profit | 751.01 | 802.81 | 970.23 | 940.11 | 984.48 | 981.84 |
| | Std | 322.59 | 306.72 | 284.87 | 274.09 | 337.89 | 311.61 |
| | Nr. Best | 7 | 7 | 37 | 29 | 56 | 64 |
| | CPU (s) | 6.03 | 6.77 | 20.8 | 5.73 | 8.51 | 8.97 |
| Instance50 | Min | −10 221.77 | −6455.03 | −8449.85 | −5976.06 | −14 129.3 | −2741.61 |
| | Max | 1033.43 | 1764.32 | 1658.34 | 1799.17 | 1740.3 | 2204.46 |
| | Avg. profit | −2162.068 | −1935.232 | −1275.51 | −830.49 | −755.156 | −512.287 |
| | Std | 1614.89 | 1606.53 | 1308.22 | 1350.35 | 1719.94 | 790.40 |
| | Nr. Best | 7 | 13 | 21 | 46 | 66 | 47 |
| | CPU (s) | 21.66 | 20.8 | 26.05 | 18.09 | 17.54 | 18.23 |
| Instance100 | Min | −42 845.88 | −38 471.87 | −46 470.4 | −30 375.6 | −71 853.7 | −22 684.2 |
| | Max | −8356.22 | −6758.12 | −2469.33 | −3923.17 | −314.55 | −1355.98 |
| | Avg. profit | −20 979.24 | −19 562.53 | −18 342.4 | −14 554.8 | −11 339.9 | −8587.88 |
| | Std | 6505.795 | 6149.1705 | 6445.754 | 5448.069 | 9445.354 | 4430.528 |
| | Nr. Best | 0 | 0 | 3 | 13 | 80 | 104 |
| | CPU (s) | 82.7 | 53.64 | 91.48 | 74.2 | 72.74 | 80.61 |

For large-sized instances (Instance100), DRL-LSH demonstrates superior scalability and performance, achieving an average profit of −8587.88, which is 13.11% better than HH-Probe, the next best-performing method. This significant improvement in performance supports findings in the literature (Kallestad et al., 2023), which highlight that DRL-based approaches tend to excel as the problem size increases due to their ability to learn effective policies for operator selection in complex and high-dimensional solution spaces. Traditional metaheuristic methods, particularly GRASP-PR and GRASP-Q, show the weakest performance, with average profits of −20 979.24 and −19 562.53, respectively. SimGRASP, while competitive in smaller instances, fails to scale effectively due to the limitations of its sample-based evaluation strategy, yielding an average profit of −18 342.4.

In terms of the number of best solutions, DRL-LSH consistently achieves the highest number across all instance sizes. In small-sized instances, it records 64 best solutions, outperforming HH-Probe (56 instances) and SimGRASP (37 instances). As the problem size increases, the advantage of DRL-LSH becomes more pronounced, achieving 47 best solutions in medium-sized instances and 104 best solutions in large-sized instances, clearly surpassing all other methods. These results highlight the robustness of DRL-LSH in providing high-quality solutions across varying problem sizes.

Regarding computational time, DRL-LSH exhibits slightly longer runtime compared to other methods, particularly HH-Q and HH-Probe. The computational time for DRL-LSH increases from 8.97 s for small-sized instances to 18.23 s and 80.61 s for medium- and large-sized instances, respectively. The slightly higher computational time of DRL-LSH can be attributed to the varying time complexities of the heuristic operators and the first-improvement local search procedure it employs. HH-Q and HH-Probe, which also employ AOS strategies, show slightly lower computational times in medium- and large-sized instances, but at the cost of reduced solution quality compared to DRL-LSH. SimGRASP exhibits the longest runtime among the methods due to its reliance on Monte Carlo sampling, which increases computational overhead.

The difference in computational time among the methods is consistent with findings from previous studies (Kallestad et al., 2023; Reijnen et al., 2024), where DRL-based heuristics, such as DRLH and DRL-ALNS, demonstrated longer computational times compared to traditional ALNS methods. However, these studies also emphasize that the increase in computational time is justified by the significant improvement in solution quality offered by DRL-based methods. Similarly, in this study, while DRL-LSH incurs slightly higher computational time, it provides consistently better solutions, particularly for larger instances where solution quality is critical.

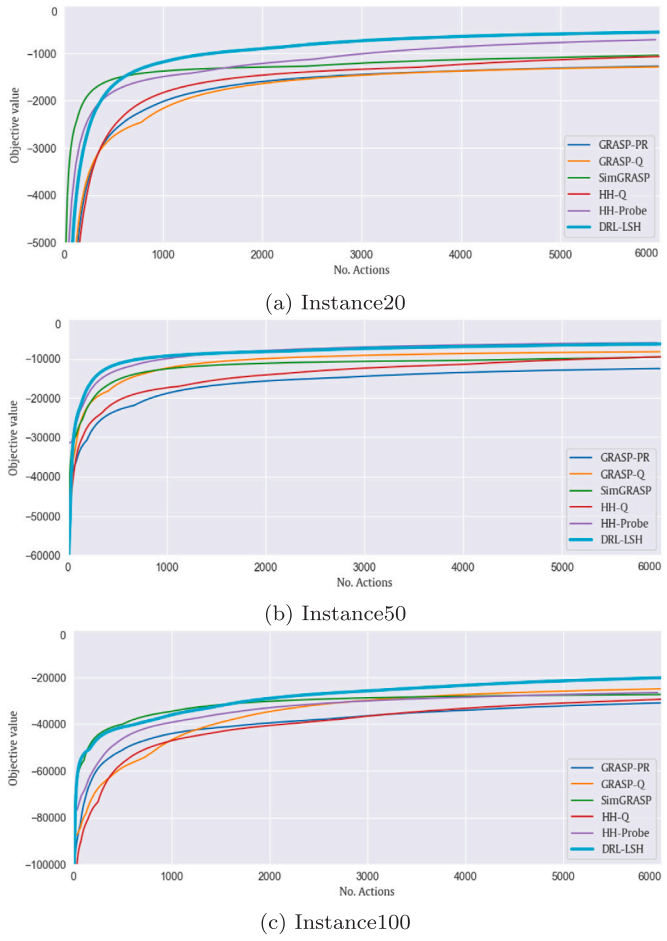
#### 5.3.2. Convergence speed

In this section, we present the convergence speed and performance characteristics. The convergence patterns, spanning over 6000 actions across three problem sizes, are illustrated in Fig. 6. Notably, DRL-LSH consistently exhibits superior performance among the considered methods across all problem instance sizes. For small-sized instances, DRL-LSH initially progresses at a slower pace compared to SimGRASP and HH-Probe before eventually achieving higher solution quality and gradually improving it. When faced with medium-sized and large-sized problems, DRL-LSH finds high-quality solutions quickly and then looks for ways to improve them before stabilizing. Due to DRL-LSH being trained to find new best solutions, its convergence line exhibits a small slope after reaching a good solution, as opposed to a straight line. This indicates its ability to continuously explore new best solutions as the number of actions increases, rather than converging prematurely to a local optimum.

#### 5.3.3. Experiment on scalability

In real-world applications, the algorithm generality is important. Generality refers to the ability of trained algorithms to perform effectively beyond the specific instances in which they were trained. Essentially, a generic model can extrapolate the patterns or relationships it has learned from its training data to new and unseen data. Deploying a DRL-based algorithm on events or additional information that did not occur during the training phase can result in poor performance (Zhang et al., 2022). In our method, DRL-LSH is trained on a single-sized problem instance, which makes it unaware of newly added nodes and vehicles. Nevertheless, we aim to observe the performance behavior of DRL-LSH when it encounters instances with significantly different characteristics compared to those used for training. Consequently, we use a set of test instances with varying numbers of nodes, ranging from 8 to 110 nodes. DRL-LSH20, 50, and 100 are used for instances with a number of nodes ranging from 8 to 24, 30 to 70, and 80 to 110, respectively. The idea of using instance size specific models comes from the existing relevant studies (Reijnen et al., 2024; Zhang et al., 2022).

The average solution quality and computational time, based on 10 runs of all methods on these 14 test instances, are reported in Table 5. It can be observed that for problems with sizes ranging from 8 to 16 nodes, SimGRASP performs the best, as Monte Carlo sampling can explore good solutions in a relatively small solution space. As the problem size increases to the range of 20 to 40 nodes, hyperheuristic methods like HH-Q and HH-Probe demonstrate their effectiveness. In these instances, DRL-LSH achieves competitive results but does not outperform the reference algorithms. However, when the problem size exceeds 50 nodes, DRL-LSH consistently performs the best among the




(a) Instance20

(b) Instance50

(c) Instance100

**Fig. 6.** Rolling average of the objective value after each action for all methods and each problem size.

methods, and the performance gap between DRL-LSH and its competitors widens as the problem size increases. The difference is attributed to the learning ability of DRL, which leverages information about the problem instance to select an appropriate sequence of heuristics. In contrast, other methods either heavily depend on domain knowledge or capture only partial information about the problem instance, which can become less effective when the structure of the problem instance changes significantly. This observation suggests that DRL-LSH exhibits strong performance even in instances for which it has no prior experience and particularly well-suited for deployment in large-scale problem instances.

### 5.3.4. Trained policies of DRL-LSH

The interpretability of complex algorithms trained through a neural network is often considered challenging. In the context of DRL-LSH, interpretability refers to the ability to understand, explain, and evaluate the decision-making process of the model, particularly how it selects local search heuristics and the rationale behind those choices during the optimization process. With DRL-LSH, heuristic operators are manually designed which enhance the interpretability of this algorithm. To gain a deeper understanding of the trained policies, we conducted experiments to investigate the behavior of the trained algorithms on the test instances. Specifically, we collected action distributions at every 300 time steps out of a total of 6000 steps for each DRL-LSH algorithm. The proportions of actions for the three DRL-LSH algorithms are presented in Fig. 7. It is evident that Action 12 — path-relinking (PR) is the most frequently chosen among the considered heuristics, followed by Action 0, which involves restarting the search process at a new point in the search space by creating a new solution. This observation is reasonable because, among intensification operators, PR is the only operator based on a population search strategy, which helps explore trajectories connecting elite/high-quality solutions.

Due to its ability to explore multiple elite neighbors, PR outperforms other considered improvement heuristics when applied once. On the other hand, Action 0 – creating a new solution – is the only diversification operator in our DRL-LSH algorithms, serving as a means to escape local optima. The learned policies of DRL-LSH on problem size 20 and 100 are relatively similar, with path-relinking dominating among other actions, occurring with a frequency of around 70% to 80% of total actions taken. In the case of problem size 50, while path-relinking still has the highest proportion, it accounts for approximately 30% to 40% of total actions taken. Additionally, the proportions of other actions, except path-relinking, are roughly similar, each comprising about 6% of the total actions. Another observation is that in the DRL-LSH trained with problem size 20 instances, where the trained instance size is small, Actions 2 - intra-relocation and 8 - or-opt (inter-route) are taken more frequently than the rest, whereas in problem size 100, the focus centers on Actions 12 and 0 to optimize the search within 6000 actions. Additionally, we can observe that actions based on intra-route neighborhood local search are not taken frequently compared to other actions. This can be attributed to the fact that local searches based on intra-moves like 2-opt (in Action 5) and Intra-swap (in Action 3) are not particularly effective in improving solutions under the time windows constraints of HVRP-MTWSTT.

To better reveal the existing patterns between the states and their corresponding actions, Fig. 8 presents the t-SNE (t-distributed Stochastic Neighbor Embedding) visualization for over 20 000 states in the search process and their associated actions within the problem size 50. It is evident that there are numerous elongated data points, primarily due to our state features comprising three sequential attributes out of a total of eight: the current iteration count, the number of consecutive iterations without reinitialization the search, and the number of consecutive iterations without improving the best-found solution. Each elongated cluster signifies a search process that combines a sequence of different operators. It is notable that the starting point of each elongated cluster corresponds to an action 0 signifying the initiation of a new solution and the commencement of a new search sequence. On the right-hand side of the t-SNE visualization, we observe combinations of Actions 0 and 12 - path-relinking, which can represent either a sequence of Action 0 followed by a series of path-relinking procedures or consecutive pairs of Actions 0 and 12. The t-SNE visualization clearly illustrates that the DRL agent employs a variety of operator sequences to search for improved solutions, highlighting the adaptability of its framework compared to fixed strategies in the search process.

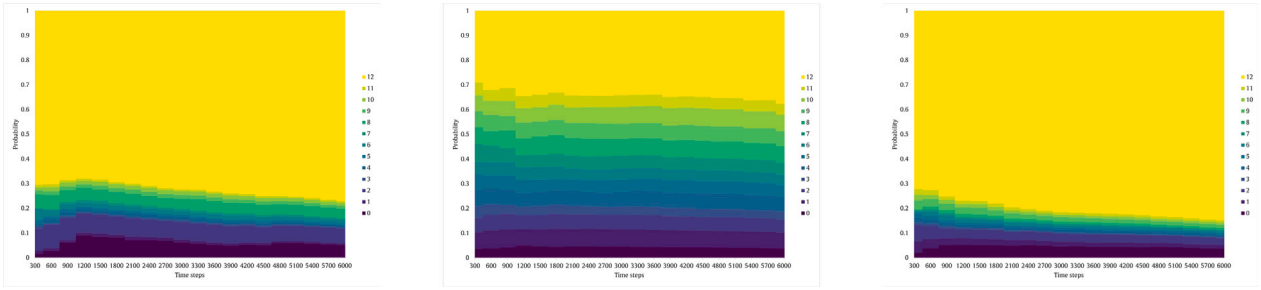
Based on the above experiment, we can gain insights into the behavior of the DRL-LSH and identify patterns that exist between the states of the search and the corresponding actions at a certain level. Decisions produced by end-to-end DRL methods are often challenging to interpret, and there are limitations in explaining their motivation. DRL-LSH, however, can provide an explainable approach at a certain level, enabling decision-makers to better understand and accept recommendations in real-world applications.

### 5.3.5. Experiment on a different reward function and statistical test

**Alternative reward function:** As discussed in Section 4.1, a well-designed reward function needs to ensure that the DRL agent learns the desired behavior effectively. The reward function should align with the ultimate objective of the problem and avoid reward hacking behavior. In this section, we conduct an experiment with an alternative reward function that directly focuses on improving the objective function of the solution found, guided by a metaheuristic, instead of encouraging exploration of new solutions. At each training step, we import the objective value of the solution of this problem instance solved by

**Table 5**
Results of scalability experiment.

| No | Instance | GRASP-PR | | GRASP-Q | | SimGRASP | | HHQ | | HHProbe | | DRL-LSH | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Avg. 10 | CPU (s) | Avg. 10 | CPU (s) | Avg. 10 | CPU (s) | Avg. 10 | CPU (s) | Avg. 10 | CPU (s) | Avg. 10 | CPU (s) |
| 1 | S–8 | 579.16 | 1.72 | 628.41 | 1.82 | **680.06** | 3.33 | 633.90 | 2.62 | 662.15 | 4.38 | 628.91 | 5.16 |
| 2 | S–12 | 120.51 | 2.99 | 157.96 | 3.09 | **325.07** | 5.57 | 192.73 | 3.68 | 295.09 | 4.70 | 240.50 | 6.45 |
| 3 | S–16 | 819.35 | 3.75 | 863.70 | 3.90 | **1099.55** | 6.35 | 856.17 | 4.00 | 999.24 | 5.65 | 939.91 | 7.55 |
| 4 | S–20 | 542.06 | 5.12 | 540.31 | 5.47 | 791.14 | 8.59 | 729.77 | 5.53 | **845.36** | 6.45 | 547.10 | 8.68 |
| 5 | S–24 | 2004.01 | 6.85 | 2000.28 | 6.82 | 2171.34 | 10.98 | 2098.93 | 6.39 | **2349.39** | 10.05 | 2270.52 | 9.96 |
| 6 | S–30 | 619.73 | 10.12 | 718.93 | 9.96 | 770.70 | 19.96 | **970.00** | 8.83 | 519.87 | 10.98 | 734.48 | 13.22 |
| 7 | S–40 | −201.27 | 15.54 | −295.29 | 13.08 | 409.94 | 20.14 | 459.41 | 11.56 | **629.12** | 13.23 | 553.39 | 12.21 |
| 8 | S–50 | −2613.90 | 23.86 | −2529.79 | 17.39 | −1065.21 | 25.16 | −1023.10 | 15.37 | −398.97 | 15.86 | **−387.21** | 17.54 |
| 9 | S–60 | −4270.13 | 36.23 | −4795.87 | 30.31 | −3682.71 | 33.56 | −1830.33 | 20.81 | −2489.74 | 18.29 | **−1000.17** | 22.23 |
| 10 | S–70 | −11 886.78 | 45.68 | −10 980.31 | 42.97 | −8295.74 | 40.91 | −8702.46 | 24.16 | −4873.71 | 21.97 | **−3055.11** | 43.54 |
| 11 | S–80 | −16 401.37 | 50.97 | −17 159.66 | 42.96 | −12 024.53 | 59.31 | −10 048.22 | 32.98 | −5806.80 | 32.77 | **−5117.24** | 45.14 |
| 12 | S–90 | −16 077.14 | 64.81 | −14 949.01 | 48.77 | −14 958.22 | 85.76 | −9656.27 | 41.88 | −13 290.74 | 46.94 | **−7596.42** | 57.14 |
| 13 | S–100 | −19 455.57 | 72.75 | −19 064.94 | 53.75 | −13 907.32 | 91.72 | −13 533.61 | 47.02 | −14 697.49 | 38.80 | **−9156.77** | 60.92 |
| 14 | S–110 | −31 065.17 | 75.52 | −28 593.94 | 85.30 | −25 965.34 | 107.73 | −22 633.78 | 58.30 | −22 264.36 | 40.70 | **−17 293.82** | 70.74 |



(a) Instance20 (b) Instance50 (c) Instance100

**Fig. 7.** Proportion of actions taken every 300 search steps for each problem size.

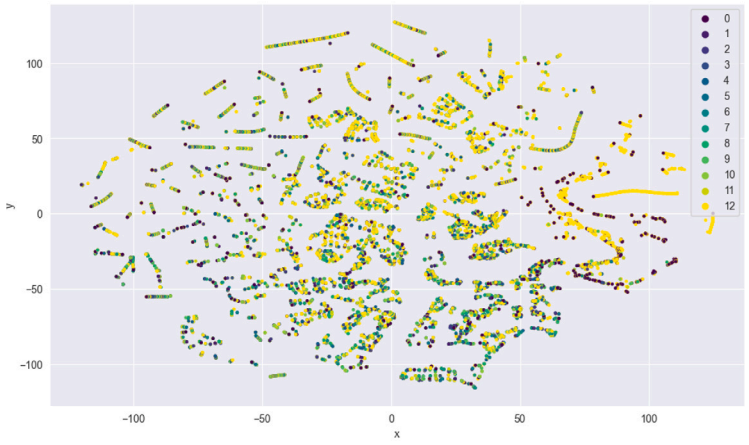


**Fig. 8.** Two-dimensional t-SNE visualization of 20,000 search states.

GRASP-PR, denoted as $f(x^*)_M$. We use this objective function as a feedback to the DRL agents about their performance.

In each time step, the DRL agent receives a reward upon reaching a predefined threshold for the first time in this training episode, e.g., 50%, 70%, or 90% of the metaheuristic's performance. With this reward strategy, there may be one action that makes the newly found solutions surpass multiple thresholds, and in this case, the DRL agent will receive all accumulated rewards in this path in one action. To prevent immediate large rewards, we safely clip rewards at a maximum of 7 units, defined as $r^{IM}(s_t, s_{t+1}) = \min(r(s_t, s_{t+1}), 7)$. When the DRL agent's action explores a new solution outperforming the solution achieved by the metaheuristic, it receives a reward $\alpha^{IM}$ for each improvement. We additionally introduce a parameter, i.e., $\sigma$, to scale the reward to an acceptable range within our reward function. The idea behind this reward function (Eq. (36)) is to encourage the DRL agent to imitate the behavior of an expert in solving a problem. This approach has the potential to make DRL-LSH converge quicker to a high-quality solution than the previous function. Furthermore, this reward function synchronizes the DRL agent's goal of finding a better solution than a high-quality one rather than solely improving the current solution.

$$r^{IM}(s_t, s_{t+1}) = \begin{cases} \alpha^{IM}, & \text{if } f(x') \text{ reaches threshold} \\ \alpha^{IM} \times \sigma^{w}, & \text{if } f(x') > f(x^*)_M \\ 0, & \text{Otherwise.} \end{cases} \tag{36}$$

Here, $f(x')$ represents the solution obtained after DRL takes an action $a_t$ at time step $t$, while $f(x^*)_M$ denotes the solution attained through a metaheuristic (i.e. GRASP-PR). Furthermore, the term $\sigma \in (0, 1)$ is introduced into the reward when the solution found by DRL-LSH

**Table 6**
Performance comparison of DRL-LSH algorithms with two different reward functions.

| Datasets | DRL-LSH with $r^{EX}$ | | DRL-LSH with $r^{IM}$ | |
|---|---|---|---|---|
| | Obj. val | CPU (s) | Obj. val | CPU (s) |
| Instance20 | 981.84 | 8.97 | **1108.65** | 13.1 |
| Instance50 | **−512.29** | 18.23 | −820.18 | 10.68 |
| Instance100 | **−8587.88** | 80.61 | −9861.74 | 176 |

surpasses the performance of the metaheuristic for $w$ times. For this reward function, we aim to encourage the DRL agent to reach different thresholds during its action trajectory. Since there are many thresholds throughout this process, only a small reward is needed to guide the DRL agent's behavior and avoid over-optimism in the actions taken. Therefore, we decide to set $\alpha^{IM} = 1$.

The threshold for rewarding the DRL agent is designed in a manner such that when the current solution on hand is poor, the next action needs to significantly improve solution quality to receive a reward. Conversely, when the current solution on hand is of high quality, an action that makes only a small improvement will still be rewarded by the agent. The details about the threshold setting are presented in the supplementary materials (Part E).

The performance comparison of DRL-LSH algorithms using two different reward functions is presented in Table 6. An intriguing observation is that the DRL-LSH algorithm employing the reward function $r^{IM}$ outperforms the one using $r^{EX}$ for problem size 20 in terms of solution quality. However, for larger instance group with 50 and 100 nodes, the DRL-LSH algorithm with $r^{IM}$ performs slightly worse than when using the $r^{EX}$ reward function. This observation may be attributed to the fact that the DRL-LSH algorithm with $r^{IM}$ is learned from the behavior of handcrafted metaheuristics, which exhibit relatively good performance in small instances. Nevertheless, the objective values of these metaheuristics do not compete favorably with those of the DRL-LSH methods in medium and large instances.

**Statistical test:** To evaluate whether the reported performance differences among different methods on the test instances are statistically significant, we performed a non-parametric Friedman test followed by a Holm post-hoc test. The Friedman test assesses whether there are overall performance differences across multiple methods when applied to the test instances, without relying on strict assumptions about the data distribution. If significant differences are detected, the Holm post-hoc test is used to compare the methods pairwise while controlling for multiple comparisons. This approach ensures that the observed differences in performance across the test instances are meaningful and not due to random variation. The details can be found in the supplementary materials (Part F). The average rankings of all algorithms are presented in Table 7. The pairwise tests on different problem sizes are depicted in Fig. 9.

DRL-LSH significantly outperforms all the competing methods on the problem instances with size 100 for a confidence level of 95%. However, the ones with the sizes of 50 and 20, the performance differences are no longer significant. This observation suggests that DRL-LSH with the reward function $r^{EX}$ performs better as the problem size increases.

#### 5.3.6. Case study

To evaluate the effectiveness of the proposed method in solving real-world, large-scale problems and to support the argument that DRL-LSH is particularly favorable for large-scale applications, we introduce a real-world case study of an e-waste collection operation in Singapore. Every year, Singapore generates approximately 60,000 tonnes of e-waste, with the rate of generation increasing annually (NEA, 2024). In 2021, the National Environment Agency (NEA) introduced a regulated e-waste management system to ensure the proper collection and handling of e-waste. The NEA appointed a government-funded corporation as the Producer Responsibility Scheme (PRS) Operator for a contract with a period of five years to manage e-waste collection across Singapore for proper treatment and recycling.

Under this scheme, PRS distributed more than 550 e-bins across Singapore as public drop-off points. These e-bins are equipped with sensors that send information about the trash levels to the system, allowing the operator to determine which e-bins need to be collected the next day. The PRS depot is located in the Tuas district, in the western region of Singapore. In addition to e-bin collection, PRS also offers doorstep collection services, where residents can request PRS staff to come to their location to collect e-waste within a specified time window on a day they are available (residents can also propose several days with different time windows). This can be arranged through an application, and a fee is charged based on the volume of e-waste collected. Due to varying road widths and e-waste size, PRS employs two types of trucks for collection tasks: the heavy-duty truck fleet uses Mercedes-Benz Arocs, while the light-duty truck fleet uses Isuzu $N$ series vehicles. For more details about the described collection scheme, we refer interested readers to the PRS and NEA Singapore websites (ALBA, 2022; NEA, 2024).

In our case study, we randomly selected 250 e-bins across Singapore and assigned them to different days within a 4-day collection period. The coordinates of the e-bins and the depot are provided by the Singapore open data portal (data.gov.sg). To simulate customer requests, we generated 250 requests based on random zip codes of residents in Singapore and assigned them to one or several days within the 4-day collection period, with random time windows varying by the day. The vehicles' original volumes are based on the dimensions of light and heavy trucks as provided by Aljohani and Thompson (2020). Operational costs are calculated based on the average income of waste truck drivers in Singapore, as provided by the Ministry of Manpower (MOM, 2024). The dimensions of e-waste items were randomly collected from a reseller website and then converted into load units. Penalty and idle costs were randomly generated. The distance matrix was calculated using the Euclidean method. Fig. 10 illustrates the location distribution of e-bins (blue circles) and customer locations (red crosses) in our case study. Detailed information about the vehicles and costs is summarized in Table 8.

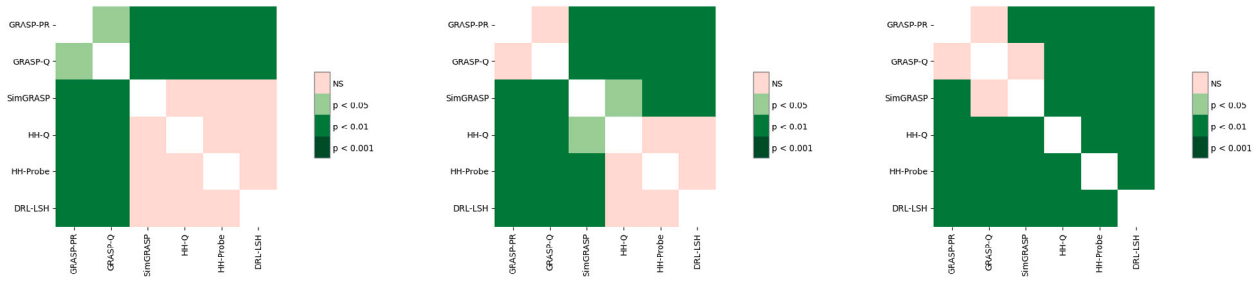
To train DRL-LSH for the case study, we generated a set of 200 synthetic instances similar to those described in Section 5.1, each with 250 e-bins and 250 customer requests, but using the cost and collection fee data from Table 8. The training rewards over time for DRL-LSH on these instances are shown in the supplementary materials (Part G). Afterwards, we deployed the trained model to solve the proposed case study and compared it with other reference algorithms. Due to the randomization in the algorithms, we also deployed all investigated methods to solve the case study 10 times, limiting the maximum computing time for each method to 5500 s. The operational profit of the best route over the 4-day collection period, as found by different algorithms, and their respective computing times, are indicated in Table 9.

Overall, the performance ranking of the algorithms remains consistent with the results obtained in the test instances featuring 100 e-bins and customers. The algorithms employing an adaptive selection operator continue to outperform those with fixed operators, while DRL-LSH confirms its strength in handling large-scale instances, achieving the best performance among all considered methods. Notably, SimGRASP now performs worse than GRASP-Q, highlighting the diminishing effectiveness of Monte Carlo sampling for finding good solutions as the instance size increases. In contrast, GRASP-Q demonstrates an advantage by partially capturing the stochastic environment through its Q-table-based tuning of operator sequences.

Closer examination of the results reveals that only HH-Probe and DRL-LSH achieve positive operational profits for the collection plan, with DRL-LSH yielding significantly higher profits, showing a 78.7% improvement over the next-best method. Regarding CPU time, as noted previously, the use of a first-improving local search strategy for each

**Table 7**
The methods average rankings.

| | GRASP-PR (i) | GRASP-Q (ii) | SimGRASP(iii) | HH-Q (iv) | HH-Probe (v) | DRL-LSH (vi) |
|---|---|---|---|---|---|---|
| Instance20 | 25.79 | 23.04 | 15.45 | 17.10 | 15.63 | **15.23** |
| Instance50 | 26.11 | 23.06 | 18.57 | 15.34 | 14.41 | **14.22** |
| Instance100 | 26.35 | 24.53 | 22.69 | 17.18 | 12.53 | **8.95** |




(a) Instance20 (b) Instance50 (c) Instance100

**Fig. 9.** Pair-wise tests of performance on different problem size.

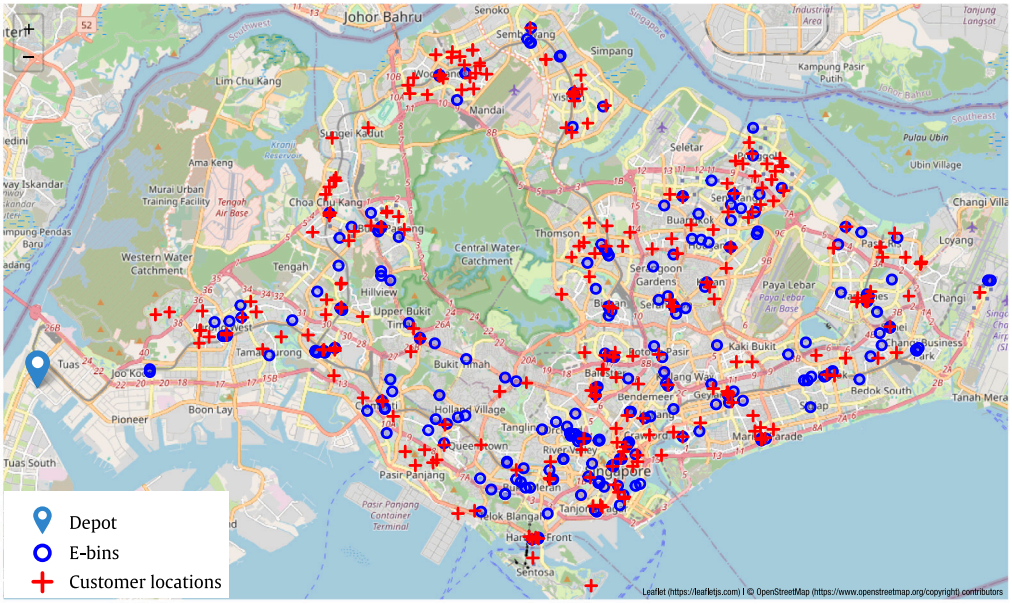



**Fig. 10.** Case study of e-waste collection in Singapore.

**Table 8**
Summary of Singapore e-waste collection case study.

| | | | |
|---|---|---|---|
| Number customers requests $N_1$ | 250 | Penalty cost of heavy trucks $\phi^1$ | $\$8 \times 10^{-2}$/time unit |
| Number of e-bins $N_2$ | 250 | Idle cost of heavy trucks $\psi^1$ | $\$4 \times 10^{-2}$/time unit |
| Number of heavy trucks $V_1$ | 30 | Capacity of lightweight trucks $q^2$ | 19 $m^3 \approx 190$ load unit |
| Number of lightweight trucks $V_2$ | 30 | Operation cost of lightweight trucks $c^2$ | $\$7 \times 10^{-2}$/time unit |
| Capacity of heavy trucks $q^1$ | $59m^3 \approx 590$ load unit | Penalty cost of lightweight trucks $\phi^2$ | $\$2 \times 10^{-2}$/time unit |
| Operation cost of heavy trucks $c^1$ | $\$12 \times 10^{-2}$/time unit | Idle cost of lightweight trucks $\psi^2$ | $\$10^{-2}$/time unit |
| | | Collection fee $p^i : i \in N_1$ | \$10/load unit |

problem-dependent heuristic leads to higher computing times for methods with superior search strategies. Consequently, the fixed-operator methods GRASP-PR and GRASP-Q terminated early, completing 6000 applications of the local search operator before reaching the maximum computation time of 5500 s. In contrast, the adaptive operator selection methods fully utilized the maximum computation time, demonstrating their ability to deploy more effective search strategies and achieve superior results.

## 6. Conclusion

We study and formulate the e-waste collection problem as the Heterogeneous Vehicle Routing Problem with Multiple Time Windows and Stochastic Travel Time (HVRP-MTWSTT). HVRP-MTWSTT encompasses various real-world constraints, including multiple-period planning, the integration of fixed drop-off collection points with customer on-demand requests within specified time windows, a heterogeneous

**Table 9**
Operational profit of best route found by investigated algorithms and the respective computing times.

| Methods | Avg. 10 ($) | CPU(s) |
|---|---|---|
| GRASP-PR | −12 044.29 | 1973.345 |
| GRASP-Q | −10 677.84 | 2003.195 |
| SimGRASP | −10 850.39 | 5500 |
| HH-Q | −2425.67 | 5500 |
| HH-Probe | 3655.45 | 5500 |
| DRL-LSH | 6531.86 | 5500 |

vehicle fleet, and stochastic travel times. We introduce a Deep Reinforcement Learning (DRL) framework to control problem-dependent local search heuristics throughout the search process, referred to as DRL-LSH. By modeling the search procedure as a sequential decision-making problem, the DRL agent can adapt its search strategies by taking actions based on problem-dependent heuristics derived from the search state.

Computational experiments demonstrate that DRL-LSH performs competitively with other metaheuristic and hyperheuristic methods on small-sized problems and outperforms these methods on large-scale instances, achieving a 24.26% improvement over the second-best method, the HH-Probe hyperheuristic. Notably, the performance gap between DRL-LSH and other methods increases as the problem size grows, stemming from the ability of DRL to capture useful information about the problem instance and drive the selection of appropriate operators. Additionally, DRL-LSH exhibits robust performance in scalability experiments, where the method is evaluated on unseen instances with additional e-bins, customers, and vehicles. The integration of low-level heuristics into the DRL framework allows DRL-LSH to provide a level of interpretability, addressing a limitation of end-to-end neural network-based methods to some extent. Moreover, we present experiments with an alternative reward function that uses the solutions generated by metaheuristics to measure performance and guide the DRL agent during the training process. In solving the Singapore e-waste collection case study with 250 e-bins and 250 customer requests, the proposed DRL-LSH method achieved an average profit value of 6531.86, outperforming the second-best method, HH-Probe, by 78.7% (3655.45) and significantly surpassing other baselines like HH-Q, SimGRASP, GRASP-Q, and GRASP-PR. These results demonstrate DRL-LSH's effectiveness in addressing large-scale, complex routing problems characterized by dynamic requests, time-sensitive tasks, and stochastic travel times, showcasing its potential for real-world applications in e-waste collection and similar domains.

For future research, it is worth considering the inclusion of more real-world stochastic elements in e-waste collection, such as stochastic service times, e-waste load variations, and other sources of uncertainty. Regarding the enhancement of the DRL-LSH algorithm, the reward function merits further experimentation in future research. Future research may also explore alternative actions for escaping local optima, such as perturbations from the current solution or the use of destroy-and-repair operators.

## CRediT authorship contribution statement

**Dang Viet Anh Nguyen:** Writing – original draft, Methodology, Formal analysis, Conceptualization. **Aldy Gunawan:** Writing – review & editing, Validation, Supervision, Methodology, Conceptualization. **Mustafa Misir:** Writing – review & editing, Validation, Methodology. **Lim Kwan Hui:** Writing – review & editing, Methodology. **Pieter Vansteenwegen:** Writing – review & editing, Validation, Methodology.

## Acknowledgement

This research was supported by the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 grant (Project ID: 21-SIS-SMU-045).

## Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.ejor.2025.04.033.

## References

Adriaensen, S., Biedenkapp, A., Shala, G., Awad, N., Eimer, T., Lindauer, M., & Hutter, F. (2022). Automated dynamic algorithm configuration. *Journal of Artificial Intelligence Research*, *75*, 1633–1699.

ALBA, E.-W. (2022). Collection Programmes – ALBA E-Waste Singapore. (Accessed 10 August 2024) https://alba-ewaste.sg/collection-programmes/.

Aljohani, K., & Thompson, R. G. (2020). An examination of last mile delivery practices of freight carriers servicing business receivers in inner-city areas. *Sustainability*, *12*(7), 2837.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D. (2016). Concrete problems in AI safety. arXiv preprint arXiv:1606.06565.

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. *Transportation Science*.

Bengio, Y., Lodi, A., & Prouvost, A. (2021). Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, *290*(2), 405–421.

Chaudhary, K., & Vrat, P. (2018). Case study analysis of e-waste management systems in Germany, Switzerland, Japan and India: A RADAR chart approach. *Benchmarking: An International Journal*, *25*(9), 3519–3540.

Di Liberto, G., Kadioglu, S., Leo, K., & Malitsky, Y. (2016). Dash: Dynamic approach for switching heuristics. *European Journal of Operational Research*, *248*(3), 943–953.

Drake, J. H., Kheiri, A., Özcan, E., & Burke, E. K. (2020). Recent advances in selection hyper-heuristics. *European Journal of Operational Research*, *285*(2), 405–428.

Festa, P., Pastore, T., Ferone, D., Juan, A. A., & Bayliss, C. (2018). Integrating biased-randomized GRASP with Monte Carlo simulation for solving the vehicle routing problem with stochastic demands. In *2018 winter simulation conference* (pp. 2989–3000). IEEE.

Fialho, Á. (2010). *Adaptive operator selection for optimization* (Ph.D. thesis), Université Paris Sud-Paris XI.

Gendreau, M., Ghiani, G., & Guerriero, E. (2015). Time-dependent routing problems: A review. *Computers & Operations Research*, *64*, 189–197.

Gendreau, M., Potvin, J.-Y., et al. (2010). *vol. 2*, *Handbook of metaheuristics*. Springer.

Glover, F. (1997). Tabu search and adaptive memory programming—advances, applications and challenges. *Interfaces in Computer Science and Operations Research: Advances in Metaheuristics, Optimization, and Stochastic Modeling Technologies*, 1–75.

Golden, B., Assad, A., Levy, L., & Gheysens, F. (1984). The fleet size and mix vehicle routing problem. *Computers & Operations Research*, *11*(1), 49–66.

Gunawan, A., Lau, H. C., & Lu, K. (2018). ADOPT: combining parameter tuning and adaptive operator ordering for solving a class of orienteering problems. *Computers & Industrial Engineering*, *121*, 82–96.

Gunawan, A., Nguyen, M. P. K., F. Yu, V., & Nguyen, D. V. A. (2023). The heterogeneous vehicle routing problem with multiple time windows for the e-waste collection problem. In *19th international conference on automation science and engineering*.

Gunawan, A., Nguyen, D. V. A., Nguyen, P. K. M., & Vansteenwegen, P. (2023). GRASP solution approach for the E-waste collection problem. In *International conference on computational logistics* (pp. 260–275). Springer.

Hemmati, A., & Hvattum, L. M. (2017). Evaluating the importance of randomization in adaptive large neighborhood search. *International Transactions in Operational Research*, *24*(5), 929–942.

Hess, C., Dragomir, A. G., Doerner, K. F., & Vigo, D. (2024). Waste collection routing: a survey on problems and methods. *Central European Journal of Operations Research*, *32*(2), 399–434.

Hildebrandt, F. D., Thomas, B. W., & Ulmer, M. W. (2023). Opportunities for reinforcement learning in stochastic dynamic vehicle routing. *Computers & Operations Research*, *150*, Article 106071.

Homberger, J., & Gehring, H. (1999). VRPTW benchmark problems. (Accessed 28 August 2023) https://www.sintef.no/projectweb/top/vrptw/homberger-benchmark/.

Kallestad, J., Hasibi, R., Hemmati, A., & Sörensen, K. (2023). A general deep reinforcement learning hyperheuristic framework for solving combinatorial optimization problems. *European Journal of Operational Research*, *309*(1), 446–468.

Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A. M., & Talbi, E.-G. (2022). Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. *European Journal of Operational Research*, *296*(2), 393–422.

Król, A., Nowakowski, P., & Mrówczyńska, B. (2016). How to improve WEEE management? Novel approach in mobile collection with application of artificial intelligence. *Waste Management*, *50*, 222–233.

Lu, H., Zhang, X., & Yang, S. (2020). A learning-based iterative method for solving vehicle routing problems. In *International conference on learning representations (ICLR 2020)*.

Martí, R., Sevaux, M., & Sörensen, K. (2024). 50 years of metaheuristics. *European Journal of Operational Research*, http://dx.doi.org/10.1016/j.ejor.2024.04.004.

Meignan, D., Schwarze, S., & Voß, S. (2016). Improving local-search metaheuristics through look-ahead policies. *Annals of Mathematics and AI*, *76*, 59–82.

MOM (2024). Occupational Wages 2023. (Accessed 10 August 2024) https://stats.mom.gov.sg/Pages/Occupational-Wages-Tables2023.aspx.

NEA, S. N. E. A. (2024). Extended Producer Responsibility (EPR) System for E-waste Management System. (Accessed 10 August 2024) https://www.nea.gov.sg/our-services/waste-management/3r-programmes-and-resources/e-waste-management.

Nguyen, D. V. A., Gunawan, A., Misir, M., & Vansteenwegen, P. (2024). Q-Learning Based Framework for Solving the Stochastic E-waste Collection Problem. In *European conference on evolutionary computation in combinatorial optimization* part of evoStar, (pp. 49–64). Springer.

Nowakowski, P., Szwarc, K., & Boryczka, U. (2018). Vehicle route planning in e-waste mobile collection on demand supported by artificial intelligence algorithms. *Transportation Research Part D: Transport and Environment*, *63*, 1–22.

Ozcan, E., Misir, M., & Burke, E. (2009). A self-organising hyper-heuristic framework. In *Proceedings of the 4th multidisciplinary international scheduling conference: theory & applications* (pp. 10–12).

Pérez-Belis, V., Bovea, M. D., & Ibáñez Forés, V. (2015). An in-depth literature review of the waste electrical and electronic equipment context: Trends and evolution. *Waste Management & Research*, *33*(1), 3–29.

Pillay, N., & Qu, R. (2018). *Hyper-heuristics: theory and applications*. Springer.

Pourhejazy, P., Zhang, D., Zhu, Q., Wei, F., & Song, S. (2021). Integrated E-waste transportation using capacitated general routing problem with time-window. *Transportation Research Part E: Logistics and Transportation Review*, *145*, Article 102169.

Rangga, U., Syed, I., Rasdi, I., Karuppiah, K., & Ikmal, I. (2019). Environmental impact, health risk, and management cost of landfilling practice: A case study in klang, selangor, Malaysia. *J. Waste Manag. Dispos*, *2*(1), 1–12.

Reijnen, R., Zhang, Y., Lau, H. C., & Bukhsh, Z. (2024). Online control of adaptive large neighborhood search using deep reinforcement learning. *vol. 34*, In *Proceedings of the international conference on automated planning and scheduling* (pp. 475–483).

Ritzinger, U., Puchinger, J., & Hartl, R. F. (2016). A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research*, *54*(1), 215–231.

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, *40*(4), 455–472.

dos Santos, J. P. Q., de Melo, J. D., Neto, A. D. D., & Aloise, D. (2014). Reactive search strategies using reinforcement learning, local search algorithms and variable neighborhood search. *Expert Systems with Applications*, *41*(10), 4939–4949.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

Serrano, B., Florio, A. M., Minner, S., Schiffer, M., & Vidal, T. (2024). Contextual stochastic vehicle routing with time windows. arXiv preprint arXiv:2402.06968.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *International conference on principles and practice of constraint programming* (pp. 417–431). Springer.

Soeffker, N., Ulmer, M. W., & Mattfeld, D. C. (2022). Stochastic dynamic vehicle routing in the light of prescriptive analytics: A review. *European Journal of Operational Research*, *298*(3), 801–820.

Szwarc, K., Nowakowski, P., & Boryczka, U. (2021). An evolutionary approach to the vehicle route planning in e-waste mobile collection on demand. *Soft Computing*, *25*(8), 6665–6680.

WEEE-Forum (2023). International E-Waste Day | WEEE Forum. (Accessed 19 September 09/19/2023) https://weee-forum.org/iewd-about/.

Zhang, Y., Bai, R., Qu, R., Tu, C., & Jin, J. (2022). A deep reinforcement learning based hyper-heuristic for combinatorial optimisation with uncertainties. *European Journal of Operational Research*, *300*(2), 418–427.

Zhang, Y., Bliek, L., da Costa, P., Afshar, R. R., Reijnen, R., Catshoek, T., Vos, D., Verwer, S., Schmitt-Ulms, F., Hottung, A., et al. (2023). The first AI4TSP competition: Learning to solve stochastic routing problems. *Artificial Intelligence*, *319*, Article 103918.

Zhang, J., Luo, K., Florio, A. M., & Van Woensel, T. (2023). Solving large-scale dynamic vehicle routing problems with stochastic requests. *European Journal of Operational Research*, *306*(2), 596–614.

Zhao, F., Zhang, L., Cao, J., & Tang, J. (2021). A cooperative water wave optimization algorithm with reinforcement learning for the distributed assembly no-idle flowshop scheduling problem. *Computers & Industrial Engineering*, *153*, Article 107082.