Learning-Aided Neighborhood Search for Vehicle Routing Problems

Tong Guo, *Graduate Student Member, IEEE*, Yi Mei, *Senior Member, IEEE*, Mengjie Zhang, *Fellow, IEEE*, Haoran Zhao, Kaiquan Cai, and Wenbo Du, *Member, IEEE*

***Abstract*—The Vehicle Routing Problem (VRP) is a classic optimization problem with diverse real-world applications. The neighborhood search has emerged as an effective approach, yielding high-quality solutions across different VRPs. However, most existing studies exhaustively explore all considered neighborhoods with a pre-fixed order, leading to an inefficient search process. To address this issue, this paper proposes a Learning-aided Neighborhood Search algorithm (LaNS) that employs a cutting-edge multi-agent reinforcement learning-driven adaptive operator/neighborhood selection mechanism to achieve efficient routing for VRP. Within this framework, two agents serve as high-level instructors, collaboratively guiding the search direction by selecting perturbation/improvement operators from a pool of low-level heuristics. Furthermore, to equip the agents with comprehensive information for learning guidance knowledge, we have developed a new informative state representation. This representation transforms the spatial route structures into an image-like tensor, allowing us to extract spatial features using a convolutional neural network. Comprehensive evaluations on diverse VRP benchmarks, including the capacitated VRP (CVRP), multi-depot VRP (MDVRP) and cumulative multi-depot VRP with energy constraints, demonstrate LaNS's superiority over the state-of-the-art neighborhood search methods as well as the existing learning-guided neighborhood search algorithms.**

***Index Terms*—Vehicle routing, variable neighborhood search, reinforcement learning, adaptive operator selection.**

I. Introduction

THE Vehicle Routing Problem (VRP) is prevalent in numerous real-world scenarios, including logistics, transportation, manufacturing, retail distribution, waste collection, and delivery planning. Over six decades of research have led to the exploration of various VRP variants that cater to real-world constraints, such as time windows, heterogeneous fleets, and multiple depots [1].

Received 21 December 2023; revised 11 September 2024; accepted 16 March 2025. Date of publication 24 March 2025; date of current version 6 June 2025. This work was supported in part by the National Natural Science Foundation of China under Grant U2333218. Recommended for acceptance by A. Niculescu-Mizil. *(Corresponding authors: Kaiquan Cai; Wenbo Du.)*

Tong Guo, Haoran Zhao, Kaiquan Cai, and Wenbo Du are with the State Key Laboratory of CNS/ATM, School of Electronic and Information Engineering, Beihang University, Beijing 100191, China (e-mail: guotong1997@buaa.edu.cn; zhaohaoran1818@buaa.edu.cn; caikq@buaa.edu.cn; wenbodu@buaa.edu.cn).

Yi Mei and Mengjie Zhang are with the Centre for Data Science and Artificial Intelligence & School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6012, New Zealand (e-mail: yi.mei@ecs.vuw.ac.nz; Mengjie.Zhang@vuw.ac.nz).

Digital Object Identifier 10.1109/TPAMI.2025.3554669

The VRP has been proven to be NP-hard [2], implying that the solution space grows exponentially as the number of customers increases. Among various approaches to solving VRP, exact methods [3], [4] guarantee solution optimality and maintain solid theoretical grounding. However, due to the NP-hardness of VRP, exact methods are only limited to solving small-scale instances and will become too time-consuming when the problem size gets large. Heuristics, on the other hand, provide high-quality solutions (often satisfactory although not optimal) at reasonable computational costs and exhibit notable generalization abilities, earning them widespread recognition among both researchers and practitioners [5]. Consequently, numerous effort has been invested over the past decades in developing vehicle routing heuristics [6].

Among the existing vehicle routing heuristics, neighborhood search stands out as one of the most extensively studied techniques due to its rapid convergence and robust exploitation capabilities. A variety of neighborhood search optimizers have been developed, including Iterated Local Search (ILS) [7], Perturbation-based Local Search (PLS) [8], Large Neighborhood Search (LNS) [9], Variable Neighborhood Descent (VND) [10], and Variable Neighborhood Search (VNS) [11], [12], each demonstrating promising results across various VRP instances and their variants. The core idea of neighborhood search is to iteratively explore and improve solutions within a local neighborhood of a given solution. Most neighborhood search optimizers utilize multiple operators to offer different kinds of moves in the search space, thus improving the connectivity and the diversification of the neighborhood [13], [14], [15]. While neighborhood search is a promising technique, its success is highly dependent on the neighborhood structure, i.e., the operators used at each iteration. The selection of operators at each iteration, along with the determination of the sequence in which these operators are employed throughout the entire search process, should be carefully designed to guarantee the success of the neighborhood search [16] (an example case is shown in Fig. 1). However, deriving such guidance rules necessitates comprehensive expert domain knowledge, which is labor-intensive and often difficult to access. Consequently, most existing neighborhood search optimizers usually exhaustively explore all considered neighborhoods with a random sequence (i.e., pre-fixed without any guidance knowledge), leading to slower convergence speed and worse ultimate performance.

Adaptive Operator Selection (AOS) is a popular technique that dynamically selects, adapts, and applies heuristic operators based on their historical performance or other feedback mechanisms during the optimization process. This approach allows

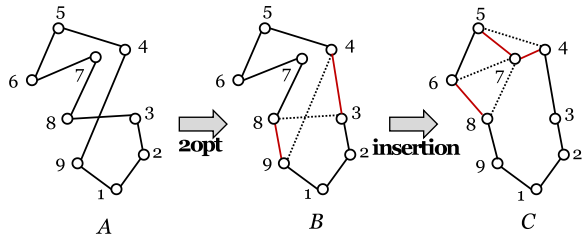


Fig. 1. Consider an example in [16] where the best operator/neighborhood for route A differs from that of its best neighbor B. For A, the optimal operator is $2opt$, which inverts the sequence between two selected nodes (3,9) and eliminates the intersection $(9-4|8-3)$. In contrast, for B, the preferred operator is $Insertion$, which selects node 4 and inserts it into the arc $(4-5)$.

for the automatic refinement of the order in which operators are executed in optimization procedures [17]. More recent efforts are integrating reinforcement learning (RL) and AOS to boost its performance [18], [19], [20], [21], [22], [23], [24]. The learning-based AOS paradigm consists of two key procedures: the first involves creating a *state representation* of the search process, and the second entails operator selection based on a *learnable policy*. After applying an operator, it receives feedback through rewards, subsequently refining its selection policy to increasingly favor operators yielding more advantageous outcomes. Compared with traditional AOS, the learning-based version not only considers immediate rewards but also learns to estimate the future utility of operators by interacting with the problem environment and incrementally updating its knowledge based on the received feedback [19].

While learning-based AOS has garnered promising results across various routing problems, most current approaches abstract the search process state representation at a macro-level, utilizing factors like objective values [18], [22], previously selected operators [20], [24], [25], and static instance characteristics [18], [23] (e.g., vehicle number and customer location in VRP). Notably, the spatial topological structure of routes (i.e., current solutions) intrinsically captures crucial details about route connectivity and proximity relations, serving as pivotal factors that directly reflect the current state and progress of the search and optimization process in VRP. However, this aspect has not been adequately considered, making existing feature representations insufficiently informative to effectively guide the search process. Additionally, most learning-based AOS only focuses on guiding the exploitation by selecting an improvement operator, ignoring the exploration during the optimization, which may get stuck in poor local optima. It is possible to simultaneously guide the selection of both an improvement operator and a perturbation operator, ensuring a balance between exploitation and exploration.

In light of these, this paper proposes a **L**earning-**a**ided **N**eighborhood **S**earch algorithm (named **LaNS**), leveraging a novel learning-based AOS approach to enhance the efficiency of searching for promising solutions in the context of VRP. Specifically, to enable effective learning of guidance knowledge, an informative state representation that provides comprehensive information for agents' decision-making is developed. The spatial structure of the routes is transformed into an image-like tensor, and a Convolutional Neural Network (CNN) is employed to capture spatial features from this representation. These spatial features are then fused with macro-level features within a multi-layer perception.

Furthermore, the paper introduces a Multi-Agent Reinforcement Learning (MARL)-guided AOS, where two agents act as high-level instructors to cooperatively guide exploitation (*search policy*) and exploration (*shaking policy*). Since the policies of these two agents interact and influence each other, conventional independent learning architectures may struggle with non-stationarity issues. To address this challenge, a knowledge-sharing mechanism distilling and sharing guidance knowledge between agents is devised. Comprehensive experiments conducted on standard capacitated VRP benchmarks (CVRP), multi-depot VRP benchmarks (MDVRP), as well as a real-world logistic VRP benchmark with more complex objectives and constraints, demonstrate that LaNS consistently outperforms state-of-the-art methods across various VRP scenarios.

The major contributions of this study are as follows:

1) We devise a new and informative state representation for routing problems, ensuring the provision of comprehensive information to facilitate optimized decision-making for neighborhood selection.
2) We develop a novel learning-aided neighborhood search algorithm, fortified with a MARL-guided neighborhood selection. This algorithm intelligently selects the most appropriate low-level heuristic based on the current search state, effectively steering and expediting the optimization process toward a (near) optimal solution.
3) We validate the effectiveness of the proposed methodology across various VRP benchmarks, showcasing effectiveness and applicability in diverse scenarios.

The remainder of this paper is organized as follows. First, the background is introduced in Section II. Then, Section III describes the proposed algorithm in detail. Experimental studies and discussions are described in Section IV. Finally, Section V draws the conclusions.

II. Background

In this section, we initially provide the formal definitions of VRP, followed by a review of existing solution algorithms, and conclude with an overview of the prevailing learning-based AOS methods.

A. Vehicle Routing Problem

The concept of VRP was first introduced by [2] in 1959. It is presented with a single depot, a set of N customers, each having a positive demand $q_i, i = 1, \ldots, N$, and a fleet with a number of V vehicles, each having a capacity Q_k. Additionally, a travel cost is defined for all pairs of points. The goal is to determine a set of routes, each originating and terminating at the depot, that minimizes the total cost, ensuring each of the N customers is visited exactly once and the cumulative demand of all customers on the route does not exceed Q_k.

VRP has been proven NP-hard [2]. Since its initial formalization, many variants have been introduced to address practical needs. These variant models can include more complex constraints, such as time windows [26], multiple depots [27], vehicle energy/battery constraints [28], etc. They can also incorporate other objective functions, such as the cumulative waiting time in emergency response scenarios [29] and fuel

consumption in environment-friendly logistics [30]. Additional constraints might increase the complexity of finding a solution since each added constraint narrows down the feasible solution space, requiring more intricate reasoning to ensure all constraints are simultaneously satisfied. In this study, we use the CVRP, MDVRP and cumulative MDVRP with energy/battery constraints (cum-MDVRP-E) as our case studies. CVRP is the most basic model, and it consists of core constraints. MDVRP can represent real-world applications with multiple garages and involves an additional decision to allocate customers to depots. cum-MDVRP-E with energy/battery constraints can reflect the recent trend of using electric vehicles in many businesses.

B. Solution Algorithms

The existing algorithms fall into three main categories: exact methods, (meta)heuristic methods, and hyperheuristic methods. Exact methods, such as dynamic programming [31], branch-and-cut [32], and column generation [33], are utilized for solving (mixed) integer programming models. While these methods assure optimal solutions, they come with a computational cost that makes them better suited for small to moderately-sized instances.

The (meta)heuristic methods, including variable neighborhood descent/search [10], [11], [12], [34], [35], adaptive neighborhood search [9], [36], [37], [38], ant colony optimization [35], [39], [40], [41] and genetic algorithms [42], [43], [44], iteratively improve upon the solutions based on certain heuristics or rules, which are often inspired by natural processes. The goal is to balance the exploration of new areas in the solution space (to avoid local minima) and the exploitation of the best-found solutions (to refine them towards optimality). Most existing (meta)heuristic methods rely on problem-specific operators to facilitate diverse moves in the search space. However, the choice of operators is often predetermined or randomized, which can limit the search.

Hyperheuristics represent a higher level of abstraction in heuristic design, focusing not on solving the optimization problems directly, but rather on efficiently selecting [45], [46] or generating heuristics [47], [48] to solve these problems. For example, Genetic Programming (GP) is utilized as a high-level strategy to select an initialization heuristic [49] or to evolve a constructive heuristic [50], [51], which in turn generates routes for VRP.

C. Learning-Based Operator Selection

Selecting the optimal operator based on the current search state can be viewed as a sequential decision-making process, involving two key steps: problem state representation and policy learning. The problem state representation aims to capture the characteristics of the problem instance or the current search state. In practice, various macro-level features, including objective values [18], [22], the most recently executed operator [20], [24], [25], and population diversity [23], have been employed for solving both continuous and combinatorial optimization problems. In contrast, micro-level features delve deeper into the current solution's structure and offer a direct reflection of the search process state. Many existing works extract the micro-level features using the direct-encoding scheme, which encodes the variables of a solution as a fixed-length vector. For instance, Ye et al. [19] adopt variable values as the state representation in continuous function optimization problems. Similarly, Rafet et al. [21] represent decision variables for knapsack problems as a binary vector (0-1). However, in scenarios like routing problems, where the solution takes the form of a route structure, direct-encoding schemes may not be applicable.

The policy serves as a mapping between the current state and the selection of an optimal operator. Throughout the optimization process, this policy is continuously refined and updated based on the gathered information. Generally, both GP and RL can be employed as techniques to learn such policies, though the majority of studies predominantly utilize RL. Various RL techniques have been employed to acquire this policy, including methods such as multi-arm bandit [20], [25], Monte Carlo tree search [52], and Q-learning [21], [22], [24]. However, in the context of many optimization tasks, which often involve thousands of iterations and diverse states, these traditional RL approaches may encounter challenges due to the high-dimensional nature of the problem. Recent advancements in this field have embraced more sophisticated RL techniques, such as the deep Q network (DQN) [18], [19], [23] and proximal policy optimization (PPO) [23], to effectively learn and approximate the policy. Unlike previous approaches that primarily focus on selecting operators for improvements, this paper aims to explore the simultaneous learning of both improvement and perturbation operators. This novel approach aims to strike a balance between exploitation and exploration strategies.

III. Learning-Aided Neighborhood Search

The newly proposed algorithm, named **L**earning-**a**ided **N**eighborhood **S**earch (LaNS), is based on the well-known Variable Neighborhood Search (VNS) [53] framework. We will first give the algorithm outline. Then, we will describe each component of the new algorithm in more detail.

A. Overall Framework

Fig. 2 depicts the overall framework of LaNS and Algorithm 1 shows the detailed procedures of LaNS. Lines 1-4 describe the initialization of LaNS. The $shaking\ policy\ \pi(\cdot|\theta_1)$ and $search\ policy\ \mu(\cdot|\theta_2)$ are randomly initiated to guide the exploration and exploitation, respectively, which are represented by neural networks with trainable parameters θ_1 and θ_2. The $counter$ is maintained to record the number of stagnant iterations. nc and nl are created to record the number of shake cycles and the number of iterations elapsed in the local search. The optimization loop comprises the perturbing phase, the local search phase, and the policy learning phase.

In lines 6–12, the perturbing phase is invoked to help jump out of local optima when the search process has been stagnant for rm cycles. Lines 7-9 calculate the rewards for the shaking policy based on the two shake-cycle-best solutions and save the experience to replay buffer. In lines 10-11, the current solution is perturbed by a perturbation operator selected by the shaking policy $\pi(\cdot|\theta_1)$ based on the current state. The state features are extracted based on the current solution and instance (elaborated in Section III-B), which provides comprehensive information on the current state to support the agents' decision-making.

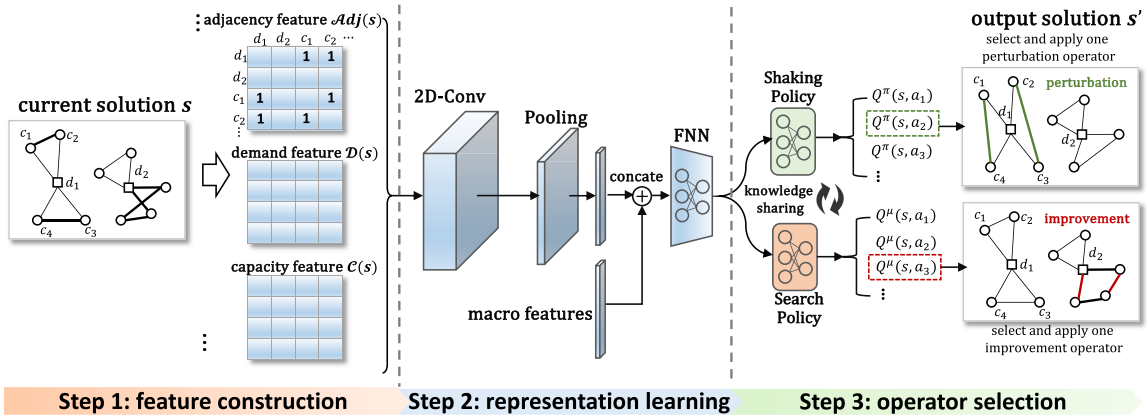


Fig. 2. The framework overview of LaNS.

Then, during the local search phase (lines 13-25), the solution is improved by an improvement operator selected by the search policy $\mu(\cdot|\theta_2)$. The improvement operator generates a list of neighbors of the current solution. If the best neighbor within the neighborhood is better than the current solution (lines 20-23), then it replaces the current solution, and the local search continues. The $counter$ is reset to zero. Conversely, if no improvement is found in this iteration, then the $counter$, increases by 1. After rm consecutive iterations without improvement, the search process is believed to be trapped in local optima, and the perturbing phase is invoked.

Finally, during the policy learning phase (lines 26-27), the two agents update their parameters based on the collected information during the search process. The learning objective of the shaking policy is to select a perturbation operator that can help jump out of local optima. In contrast, the learning objective of the search policy is to select an improvement operator that can improve the solution quality as much as possible, thus accelerating the search process. Since the state transition of the shaking policy is dependent on the search policy. Independently learning the two policies may become less effective. To solve this problem, a knowledge-sharing scheme is devised, distilling and sharing the knowledge of the search policy when training the shaking policy. The details of the policy learning (`ShakingLearn` and `SearchLearn`) are elaborated in Section III-D and III-E.

B. New Feature Construction and Representation

A novel feature tensor construction method is proposed to provide comprehensive information for agents' decision-making. Specifically, the feature tensor consists of two types of features from perspectives of instance and solution:

1) *Location feature* $\boldsymbol{\mathcal{L}}$: Its element $\mathcal{L}_{ij}$ denotes the distance between customer i and j. Thus, $\mathcal{L}$ is an instance-specific feature and retains stationary during the optimization.
2) *Adjacency feature* $\boldsymbol{Ad}(s)$: Given the current solution s (i.e., routing plan), if customer i and j are two adjacent nodes in the same route, then Ad_{ij} is 1, otherwise, it equals to 0. For example, given a route $0-c_1-c_2-0$, the adjacency matrix is defined as $Ad_{0,c_1} = Ad_{c_1,c_2} = Ad_{c_2,0} = 1$. $\boldsymbol{Ad}$ characterizes the spatial structure of a solution, thus is solution-specific.
3) *Demand feature* $\boldsymbol{\mathcal{D}}(s)$: $\mathcal{D}_{ij}$ represents the demand summation of customer i and j along the given solution s. For example, given a route $0-c_1-c_2-0$, the demand of which is [0, 2, 4, 0], then $\mathcal{D}_{0,c_1} = 2$, $\mathcal{D}_{c_1,c_2} = 6$ and $\mathcal{D}_{c_2,0} = 4$.
4) *Capacity feature* $\boldsymbol{\mathcal{C}}(s)$: $\mathcal{C}_{ij}$ describes the remaining capacity of the vehicle along the given solution s. For example, given a route $0-c_1-c_2-0$, the demand of which is [0,2,4,0] and the capacity is 10, then $\mathcal{C}_{0,c_1} = 10$, $\mathcal{C}_{c_1,c_2} = 8$ and $\mathcal{C}_{c_2,0} = 4$.
5) *Battery feature* $\boldsymbol{\mathcal{B}}(s)$: If the vehicles have additional energy or battery constraints, we utilize $\mathcal{B}_{ij}$ to characterize the remaining battery or energy of the vehicle along the given solution s. For example, given a route $0-c_1-c_2-0$, the battery cost of which is [0,1,3,0] and the battery capacity is 5, then $\mathcal{B}_{0,c_1} = 5$, $\mathcal{C}_{c_1,c_2} = 4$ and $\mathcal{C}_{c_2,0} = 1$. Note that if the vehicle energy constraints are not considered in the model, this battery feature will not be utilized in the feature tensor.

The five features are concatenated as a feature tensor $\phi(s) = \boldsymbol{\mathcal{L}} \oplus \boldsymbol{Ad}(s) \oplus \boldsymbol{\mathcal{D}}(s) \oplus \boldsymbol{\mathcal{C}}(s) \oplus \boldsymbol{\mathcal{B}}(s)$ to represent the state of the search process, which includes the information of the current solution, vehicle (capacity and battery) and customer (location and demand). Note that the feature tensor $\phi(s)$ is dependent on both the solution s and instance $\mathcal{I}$. Given a specific instance to solve, the instance-specific features do not change dynamically during the optimization process. Therefore, for simplicity in description, the feature tensor is denoted only as a variable related to the solution.

Furthermore, the multi-channel convolution neural network (CNN) is leveraged to fuse different features. The feature tensor is fed into the convolutional layer that consists of multiple filters. The k-th filter sweeps through the input feature tensor ϕ and produces:

$$h_k = RELU(\texttt{ConV2D}(W_k, \phi) + b_k) \tag{1}$$

where `ConV2D` denotes the convolution operation and the output h_k is the abstracted feature map. $RELU$ function is $RELU(x) = \max(0, x)$ and W_k and b_k are trainable parameters. Subsequent to the convolution operation, the feature maps are operated by the adaptive averaged pooling, which can allow

Algorithm 1: Learning-Aided Neighborhood Search (LaNS).

// Initialization
Initialize global best solution;
Randomly initialize the shaking policy $\pi(\cdot|\theta_1)$ and search policy $\mu(\cdot|\theta_2)$;
Set experience replay buffer $\Xi^\pi = \Xi^\mu = \emptyset$;
Set counter $counter = 0$, number of shake cycles $nc = 0$, number of iterations in local search $nl = 0$;
// Optimization Loop
while *not terminated* **do**
  // Perturb Solution for exploration
  if $counter \geq rm$ **then**
    // Calculate shaking rewards based on two recent cycle-best solutions
    $r_{nc} = 1 - \texttt{Cost}(s_{nc})/\texttt{Cost}(s_{nc-1})$;
    Construct features $\phi(s_{nc})$ and $\phi(s_{nc-1})$ based on s_{nc} and s_{nc-1};
    Save experience episode $\langle \phi(s_{nc-1}), \phi(s_{nc}), a_{nc-1}, r_{nc} \rangle$ to Ξ^π ;
    Perturbation operator selection $a_{nc} \leftarrow \pi(\phi(s_{nc})|\theta_1)$;
    Obtain new solution $s_{\text{start}} \leftarrow \texttt{Shake}(s_{nc}, a_{nc})$;
    $nc \leftarrow nc + 1$;
    Reset $counter$ to 0;
  // Local Search for exploitation
  Construct features $\phi(s_{\text{start}})$ based on s_{start} ;
  Improvement operator selection $a_{nl} \leftarrow \mu(\phi(s_{\text{start}})|\theta_2)$;
  $s_{\text{end}} \leftarrow \texttt{LocalSearch}(s_{\text{start}}, a_{nl})$;
  $r_{nl} = 1 - \texttt{Cost}(s_{\text{end}})/\texttt{Cost}(s_{\text{start}})$;
  Construct features $\phi(s_{\text{end}})$ based on s_{end};
  Save experience episode $\langle \phi(s_{\text{start}}), \phi(s_{\text{end}}), a_{nl}, r_{nl} \rangle$ to Ξ^μ;
  $nl \leftarrow nl + 1$;
  if $\texttt{Cost}(s_{end}) \leq \texttt{Cost}(s_{start})$ **then**
    Update $s_{\text{start}} \leftarrow s_{\text{end}}$;
    Update global best solution ;
    Reset $counter$ to 0;
  else
    $counter \leftarrow counter + 1$;
  // Agents learn and update
  Update shaking policy $\theta_1 \leftarrow \texttt{ShakingLearn}(\Xi^\pi)$;
  Update search policy $\theta_2 \leftarrow \texttt{SearchLearn}(\Xi^\mu)$;
return global best solution;

CNN to be more robust to the position variations in the input tensors. Finally, all feature maps are concatenated with the macro features in a fully connected neural network (FNN). Four existing macro features are utilized in LaNS, including current objective value, last executed action index, number of customers, and number of vehicles.

Compared with the existing studies, the proposed method has two major merits. First, constructing the solution-relevant features as a 2-dimensional matrix can maintain the spatial information of the solution (i.e., the route structure). Second, the multi-channel CNN model can exploit spatially local correlation of the solution structure whilst fusing the features from different perspectives.

C. Pool of Low-Level Heuristics

A rich set of low-level heuristics is used to form the action space, providing diverse moves in the search space. The details of the heuristics are presented in Table I. Most heuristics are commonly used operators for VRP or Travelling Salesman Problem (TSP), where their effectiveness has been validated in prior research. We design the pool of low-level heuristics to be as comprehensive as possible to ensure robustness across a variety of scenarios. The rationale behind using a comprehensive pool of low-level heuristics is grounded in the principle that a smaller subset of operators can be seen as a special case of a more extensive set. There are two major classes of heuristics where *intra-route* ones move customers to different positions within the same route, while *inter-route* ones move customers among different routes.

Algorithm 2: $s' \leftarrow \texttt{LocalSearch}(s, a)$.

Input: Solution s, selected neighborhood index a, neighborhood set given the solution s $\Gamma(s) = \{\mathcal{N}_1(s), \cdots, \mathcal{N}_k(s), \cdots\}$
Output: Improved solution s'
$s' = null, \texttt{Cost}(s') = \infty$;
for $s'' \in \mathcal{N}_a(s)$ **do**
  $\texttt{Cost}(s'') = \texttt{Evaluate}(s'')$;
  if $\texttt{Cost}(s'') < \texttt{Cost}(s')$ **then**
    $s' \leftarrow s''$;
    $\texttt{Cost}(s') \leftarrow \texttt{Cost}(s'')$;
return Improved solution s' ;

The local search process based on the selected operator/neighborhood a is shown in Algorithm 2. The idea is to explore the vicinity of the current solution s to find a potentially better solution. Given the current solution s and the selected neighborhood index a, each neighbor solution within the neighborhood ($s' \in \mathcal{N}_a(s)$) is evaluated. The neighborhood of a solution refers to a set of solutions that are generated based on the predefined criteria of the operators (as listed in Table I). The new solution is updated if the neighbor is better than the current solution. The loop is terminated until all the neighbors are exhausted. During the shaking phase, the current solution is intentionally perturbed using a defined shaking heuristic. This helps escape local optima by randomly sampling a neighbor s' from its current neighborhood $\mathcal{N}_k(s)$. The shaken solution then serves as a starting point for subsequent local search, enabling the algorithm to explore new regions of the solution landscape and thereby enhancing its chances of finding a global optimum.

D. MDP Formulation

The selection of heuristics by the agents is modeled as a Markov Decision Process (MDP). The optimization process is separated into a number of episodes. The agents control the optimization process online. Given the state, the agents choose an operator from the pool based on the policy and modify the current solution. The agents collect the rewards from the environment, which are calculated based on the difference between the two solutions. Then, the agents update the policy by maximizing the accumulative rewards.

The MDPs of the shaking and search policies are formulated in a similar way. The MDP is represented as a tuple with four elements $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} >$:

1) State Space $\mathcal{S}$: Given a solution s, the feature tensor $\phi(s) = \boldsymbol{\mathcal{L}} \oplus \boldsymbol{Ad}(s) \oplus \boldsymbol{\mathcal{D}}(s) \oplus \boldsymbol{\mathcal{C}}(s) \oplus \boldsymbol{\mathcal{B}}(s)$ is used to represent the state. The set of feature tensors constructed from all possible

TABLE I
DESCRIPTIONS OF THE LOW-LEVEL HEURISTICS

Class	Name	Solution generation criteria	Search	Shake
Intra-route	relocate [18]	Move m consecutive customers (m=1,2,3) to a new location.	✓	✓
	2opt [42]	Remove two edges and reconnect their endpoints in the route.	✓	
	node-node swap [36]	Exchange 2 nodes in the route.		✓
	or-opt [54]	Replace 3 arcs with 3 new arcs in the route.	✓	✓
	insertion [55]	Insert 1 node to 1 arc in the route.	✓	
Inter-route	node-node swap [36]	Exchange m nodes (m=1,2,2-inverse) with m nodes in another route		✓
	node-arc exchange [8]	Exchange 1 node with 1 arc in another route.	✓	
	cyclic-exchange [18]	Exchange cyclically customers between multiple routes.		✓
	relocate [18]	Move m consecutive customers (m=1,2) from a route to another.	✓	✓
	2opt [42]	Remove two edges and reconnect their endpoints in different routes.	✓	
	cross-change [8]	Swap 2 segments from different routes.	✓	✓
	or-opt [54]	Replace 3 arcs with 3 new arcs from another route.	✓	✓

solutions constitutes the state space $\mathcal{S}$. The shaking and search policies share the same state space.

2) Action Space $\mathcal{A}$: The low-level heuristics pools constitute the action space $\mathcal{A}$. The dimension of the action space $|\mathcal{A}|$ is determined by how many operators can be used in the pool. This study models the agents' interaction with the environment using the Q value-based method [56]. The agents select an action to execute according to their Q values via roulette-wheel selection, wherein the selected probability of action a with state $\phi(s)$ with the current solution s is given by:

$$P(a) = \frac{Q(\phi(s), a)}{\sum_{a \in \mathcal{A}} Q(\phi(s), a)}, \forall a \in \mathcal{A} \tag{2}$$

It is obvious that the action with higher Q values is more likely to be selected.

3) Transitions $\mathcal{P}$: Defined as $\mathcal{P} : \phi(s) \xrightarrow{a} \phi(s')$, the transition illustrates the shift from one state to another based on the selected action (neighborhood). This transition is intrinsically tied to the solution transformation $\Delta : s \xrightarrow{a} s'$, given the state's solution-specific nature. Specifically, the solution transformation of the search policy is defined as $\Delta_{\text{search}} : s \xrightarrow{a} s' = \texttt{LocalSearch}(s)$, i.e., the best neighbor obtained by exhaustively enumerating all the neighbors of the solution s within the neighborhood.

Conversely, for the shaking policy, its solution transformation $\Delta_{\text{shake}} : s \xrightarrow{a} s'$ is defined as follows: Given a solution s (which is a local optimum), by applying the chosen perturbation operator on this solution, a new search starting point is created. The next solution s' refers to the subsequent local optimum reached through a series of local searches starting from this new point. The transformation process is depicted in Fig. 3.

Based on the solution transformation rules of the shaking and search policies, the transition $\mathcal{P}$ is defined by the mapping $\mathcal{P} : \phi(s) \xrightarrow{a} \phi(s')$, which is stochastic due to the inherent randomness of the operators.

4) Rewards $\mathcal{R}$: Given the current solution s, and the next solution s', which are defined by the solution transformation above, the reward functions of the shaking and search policies have a unified equation:

$$r(s, a, s') = \frac{\texttt{Cost}(s) - \texttt{Cost}(s')}{\texttt{Cost}(s)} \tag{3}$$

where $\texttt{Cost}(\cdot)$ denotes the objective function of the optimization problem and it is to be minimized in our case. Thus, if the next solution s' is better (has smaller objective values) than the current solution s, then the reward has higher values. Specifically, to calculate the reward for the search policy, s denotes the current solution, and s' represents the new solution after applying the selected improvement operator on s. In contrast, the perturbation operators have long-lasting effects by affecting an entire improvement iteration. Therefore, the reward of the shaking policy is calculated by comparing the best solution in the current shake cycle s with the best solution at the end of the subsequent shake cycle (after the local search) s'. The reward of shaking policy aims to evaluate how effectively shaking generates potentially exploitable regions in the solution space, rather than immediate quality improvements. A successful shake operation, therefore, may not directly enhance solution quality but establishes a favorable starting point for future searches. Thus, the improvement ratio between successive local optima serves as a metric to evaluate how well shaking prepares for escaping local optima, aligning with its goal of diversifying and exploring new solution areas.

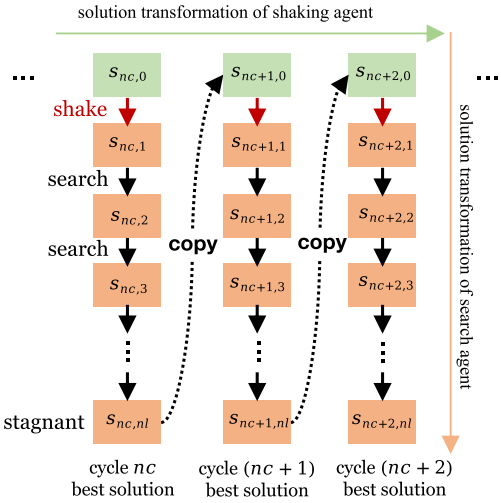


Fig. 3. Depicting the solution transformations for both shaking and search policies: The shaking policy, starting from solution $s_{nc,0}$, applies the shake operation followed by local searches, resulting in the next cycle's best solution $s_{nc,nl}(s_{nc+1,0})$. Meanwhile, the search policy takes $s_{nc,1}$, uses a chosen local search neighborhood, and transitions to an improved solution $s_{nc,2}$, representing the best neighbor within that neighborhood.

E. Policy Learning With Knowledge Sharing

1) Policy Update: The shaking and search policies cooperate with each other. For each agent, its goal is to find a policy

$\Pi : \mathcal{S} \rightarrow \mathcal{A}$, a mapping from states to actions that maximize the expected future accumulative rewards when the agent chooses actions according to Π. In the Q-value-based method, the agent iteratively improves its policy by updating its Q-function $Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ that defines the expected future accumulative reward for taking action a in state s.

Specifically, the Q value functions of the shaking and search policies are $Q^{\pi}(\cdot|\theta_1)$ and $Q^{\mu}(\cdot|\theta_2)$, which are approximated by the neural networks with trainable parameter θ_1 and θ_2, respectively. The Independent Learning (IL) architecture [57] with standard DQN learning method can be directly utilized to train the two agents by regarding them as independent agents. However, IL may be ineffective in this context since the state transition of the shaking policy is dependent on the search policy. It may be more sensible to incorporate the search policy information when the shaking agent learns its policy.

To learn the shaking policy, a knowledge-sharing mechanism is proposed. The search policy is distilled from its Q-function approximator $Q^{\mu}(\phi(s), a|\theta_2)$ [58]:

$$\xi(s) = \frac{\exp(Q^{\mu}(\phi(s), a|\theta_2))}{\sum_{a' \in \mathcal{A}} \exp(Q^{\mu}(\phi(s), a'|\theta_2))} \tag{4}$$

where $\xi(s)$ denotes the distribution of the search policy. Then, instead of only using the current state to learn the Q-function, the shaking policy learns a mapping from the current states $\phi(s)$ as well as the search policy distribution $\xi(s)$. To update the shaking policy with learnable parameter θ_1, the stochastic gradient descent (SGD) is adopted to minimize the following loss function:

$$\theta_1^* = \arg\min_{\theta_1} \mathbb{E}_{U(\Xi^{\pi})}[y - Q^{\pi}(\tilde{\phi}(s), a|\theta_1)] \tag{5}$$

$$y = r(s, a, s') + \gamma \cdot \max_{a' \in \mathcal{A}} Q^{\pi}(\tilde{\phi}(s'), a'|\theta_1^-) \tag{6}$$

where $\tilde{\phi}(s) = \phi(s)||\xi(s)$ represents the concatenation operation to combine $\phi(s)$ and $\xi(s)$ in the FNNs. $U(\Xi^{\pi})$ denotes that the samples (or minibatches) of experience are drawn uniformly at random from the relay buffer Ξ^{π}. θ_1^- denotes the previous iteration of the parameter θ_1 to help with convergence of the network. γ represents the discount factor. The details to update the shaking policy is shown in Algorithm 3.

Furthermore, the search policy is learned by the standard DQN approach. Algorithm 4 shows the overall procedure of learning the search policy. Given the experience replay buffer and the trainable parameters, the agent randomly samples a mini-batch from its memory on which to perform its Q-function updates. When estimating the Q-value for the next state, a target action-value function $\hat{Q}$ is used, which has the same architecture of Q but with delayed parameters θ^-. This approach has been shown useful to provide stable training.

Algorithm 3: $\theta_1 \leftarrow$ `ShakingLearn`(Ξ^{π}).

Input: Q function and target-Q function of shaking policy $Q^{\pi}(\cdot|\theta_1)$, $Q^{\pi}(\cdot|\theta_1^-)$ with delayed parameters θ_1^-, experience replay buffer Ξ^{π}
Output: Updated Q function $Q^{\pi}(\cdot|\theta_1)$

for *each episode* **do**
  for *each decision point* **do**
    Sample random mini-batch of experiences $\langle \phi(s), a, r(s, a, s'), \phi(s') \rangle$ in Ξ^{π};
    Calculate the search policy approximator $\xi(s)$ based on Eq. (4);
    Concatenation operation $\tilde{\phi}(s) \leftarrow \phi(s)||\xi(s)$;
    Concatenation operation $\tilde{\phi}(s') \leftarrow \phi(s')||\xi(s')$;
    Set $y = r(s, a, s') + \gamma \cdot \max_{a'} Q^{\pi}(\tilde{\phi}(s'), a'|\theta_1^-)$;
    Perform a gradient descent step on $(y - Q^{\pi}(\tilde{\phi}(s), a|\theta_1))^2$ with respect to the parameters θ_1;
    Every C steps soft-update θ_1;

return Parameters θ_1 ;

Algorithm 4: $\theta_2 \leftarrow$ `SearchLearn`(Ξ^{μ}).

Input: Q function and target-Q function of search policy $Q^{\mu}(\cdot|\theta_2)$, $Q^{\mu}(\cdot|\theta_2^-)$ with delayed parameters θ_2^-, experience replay buffer Ξ^{μ}
Output: Updated Q function $Q^{\mu}(\cdot|\theta_2)$

for *each episode* **do**
  for *each decision point* **do**
    Sample random mini-batch of experiences $\langle \phi(s), a, r(s, a, s'), \phi(s') \rangle$ in Ξ^{μ};
    Set $y = r(s, a, s') + \gamma \cdot \max_{a'} Q^{\mu}(\phi(s'), a'|\theta_2^-)$;
    Perform a gradient descent step on $(y - Q^{\mu}(\phi(s), a|\theta_2))^2$ with respect to the parameters θ_2;
    Every C steps soft-update θ_2;

return Parameters θ_2;

IV. Experimental Studies

A. Problems and Datasets

To validate the effectiveness of LaNS, three VRP variants are selected in the experiment: the Capacitated VRP (CVRP), Multi-depot VRP (MDVRP), and cumulative MDVRP with vehicle energy constraint (cum-MDVRP-E). These variants were chosen because they encapsulate a broad spectrum of real-world applications. CVRP represents fundamental capacity constraints commonly faced in transportation, and MDVRP highlights the intricacies of managing multiple distribution points. In the realm of relief logistics, the cum-MDVRP-E [8], [59], adapted from MDVRP, prioritizes the swift delivery of critical relief supplies, addressing urgent humanitarian and post-disaster recovery needs. This endeavor is crucial for saving lives. The cum-MDVRP-E aims to minimize cumulative customer waiting time, ensuring both rapid response and equitable distribution. Simultaneously, it commonly incorporates a realistic energy constraint for vehicles. Specifically, vehicle energy consumption, which varies with payload [28], is factored into the constraints to guarantee operational safety.

The experiments are carried out on the standard CVRP benchmark instances [1], standard MDVRP benchmark instances [60] as well as cumulative MDVRP benchmark instances on relief logistic routing [8]. The distribution of customers in the VRP instances includes the random, the cluster, and the random-cluster-mixed styles. For the MDVRP and cum-MDVRP-E instances, the customer and depot locations (coordinates) are generated randomly, and the distances are euclidean. The number of vehicles available for dispatching at each depot is bounded. The information and sources of the instances are publicly available.[1]

[1] http://vrp.galgos.inf.puc-rio.br/index.php/en/

TABLE II
PARAMETER SETTING OF THE PROPOSED LANS

Parameter	Returned value
Batch size for training	128
Count to invoke perturbing process	10
Learning rate	0.001
Optimizer to training neural networks	*Adam*

B. Experimental Settings

1) Competitor Algorithms: To verify the performance of LaNS on VRP, four state-of-the-art algorithms are selected as baselines: 1) **HGS**: Hybrid Genetic Search with an enhanced swap neighborhood search operator [44]. 2) **UHGS**: Unified Hybrid Genetic Search [61], [62]. 3) **ALNS**: Adaptive Large Neighborhood Search with a number of insertion and removal heuristics [9]. 4) **KGLS***: An enhanced version of Knowledge-Guided Local Search (KGLS) [37] with a self-adaptive neighborhood size control scheme [38].

For MDVRP, the three state-of-the-art baselines are: 1) **CoEA**: Coevolution Evolutionary Algorithm with divide-and-conquer strategy for MDVRP [63]. 2) **MDALNS**: Adaptive Large Neighborhood Search for VRP with multiple depots [9]. 3) **VNSALS**: Variable Neighborhood Search with Adaptive Local Search [12].

Finally, LaNS is compared with four competing algorithms on the relief logistics routing problem. 1) **POPMUSIC**: A divide-and-conquer matheuristic approach developed by Lalla-Ruiz and Voß [64], [65], which has achieved promising results on many routing tasks. The implemented version is given in [66], which solves a similar routing problem with multi-depots and cumulative objectives. 2) **PLS**: Perturbation-based Local Search for multi-depot VRP with cumulative objectives under post-disaster scenarios [8]. 3) **VND-TSH**: Variable Neighborhood Descend with a Tabu Search Heuristic, wherein the Tabu search is utilized to avoid repetitious exploration [10]. 4) **DMMAS**: A Max-Min ant system for multi-depot VRP with cumulative objectives [67].

2) Parameter Setting: For all the compared algorithms, we adopt the parameter values suggested in their papers, presuming that they have already been fine-tuned. The parameters of LaNS are tuned using IRACE [68]. The parameter combination with the best performance is reported in Table II. The proposed LaNS algorithm is a new online AOS method, where the online policy learning might incur additional computational overhead compared to the approaches without online learning. To ensure fair comparisons, the same CPU runtime is given to all algorithms as the stopping criterion. For the neural network architecture of LaNS, the CNN module has two hidden layers, wherein the first hidden layer convolved eight 7×7 filters with stride 1 and applies the rectified linear unit ($RELU$), followed by the adaptive averaged pooling operation. The second hidden layer convolved eight 3×3 filters with stride 1. The CNN is followed by a single fully connected layer mapping the feature maps to 128 hidden nodes. The two policy networks have the same architecture: a two-layer fully connected neural network with a softmax activation output layer. The first layer has 64 hidden units, and the second layer has 32 hidden units.

Each algorithm is executed 30 times independently on each instance. All the experiments are conducted on a server running a 64-bit version of Windows 10 Pro. The hardware configuration of the host consists of an Intel(R) Xeon(R) Gold 6240R CPU with 64 GB of RAM and 4 GPUs of NVIDIA GeForce RTX 3090. All code is implemented using a combination of Python 3.8 and Java SE 17. PyTorch is employed for constructing the high-level reinforcement learning module, while Java handles the low-level heuristics.

C. Results and Discussions

First, we evaluate the objective values obtained by various algorithms on the test benchmarks. For both the standard CVRP and MDVRP, the current Best-Known Solution (BKS) has been publicly reported. Thus, we utilize the average gaps to BKS as the evaluation metrics. The gap to BKS is calculated as:

$$\text{Gap} = \frac{\text{cost}_{\text{algorithm}} - \text{cost}_{\text{BKS}}}{\text{cost}_{\text{BKS}}} \tag{7}$$

A smaller gap value signifies superior performance. A negative gap value implies that the cost determined by the algorithm outperforms the BKS. The average gaps to BKS from 30 repetitive runs of all compared algorithms on CVRP and MDVRP are detailed at the bottom of Tables III and IV, respectively.

In summary, LaNS demonstrates superior performance over other baseline methods on the CVRP and MDVRP benchmarks. Specifically, for the CVRP, the result from LaNS's 30 repeated runs exhibits a small average gap (0.17%) compared to baseline methods. In addition, the statistical significance test shows that LaNS is significantly better than all the compared baselines across most instances.

In the MDVRP benchmark, from 30 independent runs, we can observe that the average gap to BKS for LaNS is smaller than that of the second best baseline CoEA (i.e., 0.22% compared to 0.46%). The statistical significance test also indicates that LaNS has better performance than CoEA on more instances (3 out of 12).

Regarding cum-MDVRP-E, a real-world application, there have been no best-known solutions reported to date. The comparison results on the testing instances are shown in Table V, including the mean value, the standard deviation of the objective function (i.e., the total sum of arrival times at the customers), and the significance test result. The Wilcoxon Rank-Sum Test with a significance level of 0.05 and Bonferroni correction is used for the significance test. From the results, it is clear that LaNS significantly outperformed all the compared algorithms across most instances.

D. Overhead of Online Policy Update

The proposed LaNS algorithm is a new online AOS method, which does not need an offline training process. However, the online AOS may incur additional computational overhead compared to other approaches without online learning, due to the policy training/update during runtime. To evaluate this overhead, we analyze the computational complexity of each component within LaNS at each cycle. As depicted in Algorithm 1, the primary complexities in LaNS arise from `LocalSearch` in line 16 (further detailed in Algorithm 2) and the online policy update, `ShakingLearn`(Ξ^{π}) and `SearchLearn`(Ξ^{μ}), in lines

TABLE III
MEAN GAP TO THE BEST-KNOWN SOLUTIONS OF THE 30 RUNS OF THE COMPARED ALGORITHMS ON CVRP

Instance	BKS	HGS	UHGS	KGLS*	ALNS	LaNS
X-n344-k43	42050	**42075.6** (=)	42086.0 (=)	42331.6 (39.0)(+)	42253.3 (11.7)(+)	42083.2 (19.3)
X-n351-k40	25896	25943.6 (+)	25972.8 (+)	26189.8 (53.9)(+)	26275.8 (54.0)(+)	**25938.8** (23.4)
X-n359-k29	51505	**51620.0** (=)	51653.8 (+)	51863.5 (29.4)(+)	51722.7 (68.6)(+)	51632.9 (25.5)
X-n367-k17	22814	**22814.0** (=)	**22814.0** (=)	22947.2 (20.3)(+)	23090.1 (12.6)(+)	22819.5 (8.4)
X-n376-k94	147713	**147714.5** (=)	147719.0 (=)	147936.6 (14.8)(+)	147946.4 (25.5)(+)	**147714.5** (8.8)
X-n384-k52	65940	66049.1 (+)	66163.7 (+)	66429.4 (124.5)(+)	66671.3 (136.2)(+)	**66008.9** (52.9)
X-n393-k38	38260	**38260.0** (=)	38281.4 (=)	38486.4 (40.8)(+)	38703.2 (39.4)(+)	38296.4 (19.8)
X-n502-k39	69226	69239.5 (+)	69300.8 (+)	69366.6 (7.3)(+)	69352.9 (18.0)(+)	**69229.4** (7.3)
X-n513-k21	24201	**24201.0** (=)	24206.5 (=)	24354.3 (10.1)(+)	24492.9 (14.5)(+)	24205.7 (8.2)
X-n524-k153	154593	154747.6 (+)	154890.1 (+)	155673.7 (167.3)(+)	154703.4 (172.0)(=)	**154607.9** (73.6)
X-n536-k96	94868	95091.9 (+)	95205.1 (+)	95883.1 (173.6)(+)	96794.5 (324.6)(+)	**94903** (111.8)
X-n548-k50	86700	**86778.4** (=)	86970.8 (+)	86979.3 (45.0)(+)	86875.4 (66.7)(+)	86780.9 (50.4)
X-n561-k42	42717	42742.7 (=)	42783.9 (+)	43033.6 (26.7)(+)	43184.3 (17.7)(+)	**42741.1** (13.1)
X-n783-k48	72394	**72790.7** (-)	73027.6 (+)	73145.1 (77.6)(+)	73281.7 (105.1)(+)	72938.4 (130.8)
X-n837-k142	193737	**194231.3** (-)	194636.5 (=)	195091.3 (102.2)(+)	197314.2 (131.7)(+)	194632.7 (102.5)
X-n876-k59	99299	**99682.7** (-)	99889.4 (=)	100146.4 (78.3)(+)	100200.2 (83.6)(+)	99863.2 (68.0)
W/T/L	/	5/8/3	9/7/0	16/0/0	15/1/0	/
Mean Gap	/	0.146%	0.259%	0.677%	0.924%	0.170%

The Wilcoxon rank-sum test with Bonferroni correction is conducted where the significance level is set to 0.05. (+) represents LaNS is significantly better than the compared algorithm. (=) represents that there is no significant difference between LaNS and the compared algorithm. (−) denotes LaNS is significantly worse than the compared algorithm. "W/T/L" represents on how many instances LaNS wins/ties/loses the comparison.
The best results for each instance are highlighted in bold. The reported significance test and mean gap are derived from all instances. The full results on all the instances can be found in the supplementary material.

TABLE IV
MEAN GAP TO THE BEST-KNOWN SOLUTIONS OF THE 30 RUNS OF THE COMPARED ALGORITHMS ON MDVRP

Instance	BKS	CoEA	VNSALS	MDALNS	LaNS
P01	576.87	**576.87** (=)	579.0 (8.3)(=)	585.0 (16.2)(+)	577.2 (8.0)
P02	473.53	475.06 (=)	476.0 (11.9)(=)	478.0 (6.8)(=)	**473.8** (10.0)
P03	641.19	**643.57** (=)	650.0 (6.2)(+)	649.0 (9.5)(+)	645.4 (4.3)
P04	1001.59	1011.42 (+)	1017.0 (13.3)(+)	1008.0 (4.5)(+)	**1001.6** (9.9)
P05	750.03	752.39 (=)	756.0 (4.7)(=)	756.0 (3.0)(=)	**752.0** (8.2)
P06	876.5	**877.86** (=)	888.0 (6.3)(+)	883.0 (9.8)(=)	879.2 (14.9)
P07	885.8	893.36 (=)	895.0 (7.4)(=)	**889.0** (8.4)(=)	890.3 (8.6)
P08	4437.68	4474.23 (+)	4577.0 (12.1)(+)	**4421.0** (19.6)(-)	4449.5 (27.1)
P09	3900.22	3904.92 (=)	3987.0 (15.2)(+)	**3892.0** (17.7)(-)	3906.3 (14.7)
P10	3663.02	3680.02 (=)	3774.0 (25.5)(+)	**3666.0** (21.3)(-)	3681.0 (23.4)
P11	3554.18	3593.37 (+)	3568.3 (6.3)(+)	3609.3 (26.3)(+)	**3548.2** (14.5)
P12	1318.95	**1318.95** (-)	1338.5 (8.7)(+)	1333.6 (7.0)(+)	1319.0 (8.9)
W/T/L	/	3/8/1	8/4/0	5/4/3	/
Mean Gap	/	0.46%	1.44%	0.69%	0.22%

The best results for each instance are highlighted in bold. The reported significance test and mean gap are derived from all instances. The reported significance test is derived from all instances. The full results on all the instances can be found in the supplementary material.

TABLE V
MEAN, STANDARD DEVIATION AND SIGNIFICANCE TEST OF THE 30 RUNS OF THE COMPARED ALGORITHMS ON CUM-MDVRP-E

Instance	PLS	DMMAS	POPMUSIC	VND-TSH	LaNS
P14_80_2	6391.3 (305.62)(+)	6576.8 (244.21)(+)	6155.1 (299.16)(+)	6135.7 (195.16)(+)	**5639.9** (87.12)
P15_160_4	9654.8 (228.08)(+)	10866.3 (386.4)(+)	11014.9 (112.94)(+)	9283.2 (231.22)(=)	**8932.7** (75.28)
P16_160_4	10115.7 (197.6)(+)	11573.5 (429.65)(+)	11825.2 (148.02)(+)	9775.7 (165.19)(+)	**9287.4** (78.31)
P17_160_4	10714.3 (294.77)(+)	12746.3 (583.5)(+)	13012.1 (159.47)(+)	10388.0 (348.73)(+)	**9863.2** (316.37)
P18_240_6	14677.5 (562.56)(+)	20433.6 (655.41)(+)	18176.0 (408.58)(+)	14070.4 (415.53)(+)	**13226.5** (98.76)
P19_240_6	15864.9 (494.94)(+)	21539.5 (997.31)(+)	19040.6 (447.66)(+)	15300.9 (506.55)(+)	**14158.3** (206.56)
P20_240_6	17172.1 (624.32)(+)	21976.8 (950.48)(+)	20458.3 (289.84)(+)	16534.0 (624.01)(+)	**15058.8** (176.64)
P21_360_9	21895.8 (545.51)(+)	34050.4 (1409.84)(+)	28423.9 (693.47)(+)	21075.8 (390.44)(=)	**19881.1** (153.83)
P22_360_9	24699.5 (575.75)(+)	35334.4 (857.83)(+)	31395.5 (479.09)(+)	23743.5 (458.12)(+)	**22174.6** (360.85)
P23_360_9	28923.9 (703.35)(+)	42047.6 (1113.59)(+)	39064.3 (797.94)(+)	27434.6 (598.75)(+)	**25735.0** (302.19)
W/T/L	10/0/0	10/0/0	10/0/0	8/2/0	/

The best results for each instance are highlighted in bold. The full results on all the instances can be found in the supplementary material.

TABLE VI
PERCENTAGE OF RUNTIME OF THE ONLINE POLICY LEARNING IN THE ENTIRE ALGORITHM RUNTIME

Problem	Total	$< 1\%$	$[1\%, 5\%]$	$> 5\%$
CVRP	100	77	20	3
MDVRP	33	8	17	8
cum-MDVRP-E	33	15	13	5

27-28 (elaborated in Algorithms 4-5). For a given solution with N nodes, the neighborhood size generated by an operator selected from the pool in Table I is bounded by the *cross-change* operator, which is $O(N^4)$. Evaluating a neighboring solution requires $O(N)$. Therefore, the total complexity of local search at each cycle is $O(N^5 \cdot Move\_Num)$ in each cycle, where $Move\_Num$ is the number of movements executed. As for the CNN-based policy update, the overall complexity is $O(N^2\, K^2 C_{in} C_{out} \cdot Sample\_Num)$ where K is the kernel size, C_{in} and C_{out} represent the number of input and output channels, and $Sample\_Num$ is the number of traning samples used. Since $N \approx Move\_Num > Sample\_Num \gg K \approx C_{in} \approx C_{out}$, we can observe the overall complexity for each cycle of LaNS is mainly dominated by the local search operation. In addition, the proportion of computational overhead resulting from policy learning operation in the entire algorithm complexity will be reduced when the problem scale increases.

To further evaluate whether the training requires enormous amounts of time or not, we conduct experiments to record the percentage of runtime dedicated to the online policy update compared to the total runtime of the LaNS algorithm across various instances. The distribution of the overhead ratio across different test problems within varying percentage ranges is shown in Table VI. From the results, we can observe that the overhead of the online policy update in LaNS can be negligible (less than 5% of the total computational cost) in most cases.

V. FURTHER ANALYSIS

A. Ablation Study

To demonstrate the effectiveness of different components of LaNS, we compare LaNS with its four variants. Four variants of LaNS are devised for the ablation study. 1) The MARL agent-based operator selection mechanism is removed from LaNS (named **LaNS/OS**). It becomes a classic VNS that selects neighborhoods randomly without any guidance. 2) The search process is only guided by the shaking policy (named **LaNS-shake**). 3) The search process is only guided by the search policy (named **LaNS-search**). 4) The search and shaking policies independently learn the guiding knowledge without the knowledge sharing mechanism (named **LaNS-independent**).

The results, including the mean, the standard deviation of the objective values, and the significance tests, are reported in Table VII. Analogous to (24), we can determine the performance gap between various algorithms and LaNS. A positive gap indicates that LaNS is better to the algorithm being compared. In comparison to LaNS/OS, LaNS outperforms significantly on most instances, highlighting the substantial impact of knowledge guidance in enhancing overall performance. Moreover, when compared to LaNS-shake and LaNS-search, LaNS consistently attains superior results across most instances, underscoring the effectiveness of both the shaking and search policies in performance improvement. Notably, given the slightly superior performance of LaNS-Search over LaNS-Shake, it suggests that the search policy may contribute more significantly to overall performance enhancement. Finally, it shows that removing the knowledge-sharing mechanism from LaNS (i.e., two agents learn independently) leads to performance degeneration. While the significance test indicates the substantial difference between LaNS-independent and LaNS is small, LaNS often yields a smaller objective value than LaNS-independent across the majority of instances. The results validate the effectiveness of the knowledge sharing mechanism in enhancing the cooperation between the two agents.

B. Effectiveness of Learning-Aided Scheme

In this subsection, LaNS is further compared with the existing AOS and state representations to verify its effectiveness.

1) Comparison With Existing AOS: Two recently proposed learning-based operator selection methods are selected as baselines: 1) **PPO** [23]: A learning-based operator selection method for capacitated VRP with a time window, in which eight problem features are directly encoded as the state vector. The Proximal Policy Optimisation (PPO) with a fully connected neural network is trained to guide the search direction by selecting a specific operator. 2) **DQN** [18]: A learning-based operator selection method guided by deep Q agent (DQN) for VRP. DQN has ten problem features that are encapsulated as a vector. **LaNS-**F_{PPO} and **LaNS-**F_{DQN} are two variants derived from PPO and DQN, respectively. In these variants, the traditional single-agent operator selection is replaced with the proposed multi-agent operator selection scheme, while the feature representations remain unchanged. The comparison results are presented in Table VIII.

Overall, LaNS consistently outperforms the compared AOS methods in the majority of instances. Comparing PPO with LaNS-F_{PPO} and DQN with LaNS-F_{DQN}, we notice that when using the same feature representations, the proposed MARL-based operator selection framework can outperform a single-agent counterpart. This indicates that MARL can effectively guide the optimization process in selecting both improvement and perturbation operators, thereby achieving a better balance between exploitation and exploration. In contrast, PPO and DQN solely focus on selecting an operator to reduce the objective value, even when the search process is trapped in local optima.

2) Comparison With Existing State Representations: To verify the advantages of the newly proposed feature presentations, four variants of PPO, DQN and LaNS are created: 1) **PPO**+: The feature representation of PPO (F_{PPO}) is replaced with the proposed features of LaNS. 2) **DQN**+: The feature representation of DQN (F_{DQN}) is replaced with the proposed features of LaNS. **LaNS-**F_{PPO} and **LaNS-**F_{DQN} utilize the same operator selection scheme as LaNS but incorporate the feature representations from PPO and DQN, respectively. The comparison results are presented in Table VIII. As shown in the results, by equipping the newly devised feature representations, PPO+ and DQN+ achieve better results than their counterparts PPO and DQN on most instances. Additionally, LaNS generally outperforms

TABLE VII
MEAN, STANDARD DEVIATION AND SIGNIFICANCE TEST OF THE 30 RUNS OF THE COMPARED LANS VARIANTS

Instance	LaNS/OS	LaNS-shake	LaNS-search	LaNS-independent	LaNS
X-n344-k43	42364.2 (14.5)(+)	42302.3 (24.9)(+)	42121.1 (16.0)(+)	42088.2 (23.2)(=)	42083.2 (19.3)
X-n351-k40	25981.4 (39.3)(+)	25947.4 (33.2)(=)	25944.4 (36.9)(+)	25937.6 (36.7)(=)	25938.8 (23.4)
X-n359-k29	51837.4 (18.4)(+)	51774.3 (27.6)(+)	51636.0 (31.8)(=)	51645.1 (35.1)(=)	51632.9 (25.5)
X-n367-k17	22870.3 (14.2)(+)	22838.1 (12.0)(+)	22835.8 (13.3)(+)	22828.7 (13.3)(=)	22819.5 (8.4)
X-n376-k94	147973.2 (14.9)(+)	147744.4 (12.6)(+)	147734.4 (14.0)(+)	157579.8 (13.9)(+)	147714.5 (8.8)
X-n384-k52	66150.1 (88.9)(+)	66091.0 (75.2)(+)	66042.0 (83.5)(=)	66041.0 (83.0)(=)	66008.9 (52.9)
X-n393-k38	38363.5 (33.4)(+)	38309.9 (28.2)(=)	38300.4 (31.3)(=)	38296.3 (31.2)(=)	38296.4 (19.8)
X-n502-k39	69354.0 (12.3)(+)	69280.9 (10.4)(+)	69246.6 (11.5)(+)	69238.9 (11.4)(=)	69229.4 (7.3)
X-n513-k21	24246.9 (13.9)(+)	24233.8 (11.7)(+)	24209.7 (13.0)(=)	24206.0 (13.0)(=)	24205.7 (8.2)
X-n524-k153	154985.4 (123.8)(+)	154856.7 (104.6)(+)	154760.5 (116.2)(+)	154725.8 (115.6)(=)	154607.9 (73.6)
P01	589.0 (1.7)(+)	588.3 (6.2)(+)	575.3 (9.8)(=)	578.2 (12.8)(=)	577.2 (8.0)
P02	510.7 (5.0)(+)	488.0 (9.3)(+)	483.1 (14.8)(=)	481.8 (9.6)(=)	473.8 (10.0)
P03	669.4 (8.3)(+)	652.2 (10.8)(+)	649.3 (16.1)(=)	643.3 (9.4)(=)	645.4 (4.3)
P04	1070.5 (16.0)(+)	1015.9 (22.4)(+)	1015.5 (28.8)(+)	1009.4 (9.8)(+)	1001.6 (9.9)
P05	775.4 (10.2)(+)	766.6 (18.7)(+)	758.9 (8.6)(+)	753.9 (5.5)(=)	752.0 (8.2)
P06	915.8 (5.5)(+)	887.0 (19.5)(=)	893.3 (9.7)(+)	883.4 (7.8)(=)	879.2 (14.9)
P07	922.5 (9.2)(+)	894.7 (16.6)(=)	892.5 (17.4)(=)	886.9 (17.2)(=)	890.3 (8.6)
P08	4471.2 (13.8)(+)	4445.2 (18.0)(=)	4450.5 (10.0)(=)	4450.6 (9.5)(=)	4449.5 (27.1)
P09	3943.6 (25.3)(+)	3918.5 (9.7)(+)	3917.5 (15.7)(=)	3914.4 (23.9)(=)	3906.3 (14.7)
P10	3718.4 (15.1)(+)	3683.6 (12.0)(=)	3676.6 (18.2)(=)	3688.6 (18.8)(=)	3681.0 (23.4)
P14_80_2	6266.6 (176.5)(+)	5853.7 (8.8)(+)	5675.2 (100.7)(=)	5702.2 (93.6)(+)	5639.9 (87.12)
P15_160_4	10675.6 (526.9)(+)	8996.4 (52.2)(=)	8964.3 (60.5)(=)	8997.4 (102.2)(=)	8932.7 (75.28)
P16_160_4	10960.9 (353.2)(+)	9344.8 (102.7)(+)	9408.8 (90.3)(+)	9290.2 (74.1)(=)	9287.4 (78.31)
P17_160_4	11515.3 (216.4)(+)	10009.7 (294.7)(+)	9756.0 (88.1)(=)	9967.8 (120.6)(=)	9863.2 (316.37)
P18_240_6	16227.0 (421.0)(+)	13507.1 (167.7)(+)	13455.0 (155.6)(+)	13486.8 (104.2)(+)	13226.5 (98.76)
P19_240_6	16976.0 (479.0)(+)	14232.0 (216.8)(=)	14368.2 (109.5)(+)	14388.1 (107.7)(+)	14158.3 (206.56)
P20_240_6	18405.6 (536.6)(+)	15993.2 (624.7)(+)	15084.8 (401.8)(=)	15337.0 (949.0)(+)	15058.8 (176.64)
P21_360_9	24833.7 (852.2)(+)	20283.2 (233.1)(+)	20244.9 (113.0)(+)	19976.5 (99.6)(+)	19881.1 (153.83)
P22_360_9	27660.6 (766.1)(+)	22583.5 (260.9)(+)	22512.3 (207.9)(+)	22230.5 (186.8)(=)	22174.6 (360.85)
P23_360_9	32310.8 (1336.0)(+)	25710.9 (354.6)(=)	25711.0 (353.1)(=)	25709.4 (152.9)(=)	25735.0 (302.19)
W/T/L	30/0/0	21/9/0	14/16/0	7/23/0	/

The best results for each instance are highlighted in bold. The reported significance test is derived from all instances. The full results on all the instances can be found in the supplementary material.

TABLE VIII
MEAN, STANDARD DEVIATION AND SIGNIFICANCE TEST OF THE 30 RUNS OF THE COMPARED AOS AND STATE REPRESENTATIONS

Instance	PPO	DQN	PPO+	DQN+	LaNS-F_{PPO}	LaNS-F_{DQN}	LaNS
X-n344-k43	42171.1 (14.6)(+)	42131.7 (22.1)(+)	42150.8 (19.8)(+)	42137.9 (33.7)(+)	42136.4 (23.1)(+)	42130.4 (18.3)(+)	42083.2 (19.3)
X-n351-k40	25950.3 (33.8)(=)	25944.4 (36.3)(+)	25933.3 (33.2)(=)	25933.7 (41.9)(=)	25938.5 (36.9)(=)	25948.8 (40.3)(=)	25938.8 (23.4)
X-n359-k29	51701.7 (28.1)(+)	51749.6 (45.8)(+)	51684.0 (16.2)(+)	51704.0 (25.3)(+)	51663.9 (15.1)(+)	51724.9 (23.5)(+)	51632.9 (25.5)
X-n367-k17	22842.0 (12.2)(+)	22826.9 (13.1)(=)	22834.9 (12.0)(=)	22823.9 (15.2)(=)	22825.8 (13.4)(=)	22812.1 (14.6)(-)	22819.5 (8.4)
X-n376-k94	147741.2 (12.8)(+)	147784.0 (13.7)(+)	153470.3 (12.6)(+)	157790.4 (15.9)(+)	147754.3 (14.0)(+)	147755.8 (15.3)(+)	147714.5 (8.8)
X-n384-k52	66045.3 (76.6)(=)	66039.0 (82.1)(=)	66043.2 (75.1)(+)	66032.7 (94.8)(=)	66047.6 (83.6)(+)	66013.5 (91.2)(=)	66008.9 (52.9)
X-n393-k38	38302.9 (28.7)(=)	38319.8 (30.8)(+)	38311.5 (28.2)(+)	38288.3 (35.6)(=)	38302.3 (31.4)(=)	38291.2 (34.2)(=)	38296.4 (19.8)
X-n502-k39	73013.7 (10.6)(+)	70395.6 (11.3)(+)	69257.2 (10.4)(+)	69243.7 (13.1)(=)	69249.9 (11.5)(+)	74409.0 (12.6)(+)	69229.4 (7.3)
X-n513-k21	24219.8 (11.9)(+)	24207.1 (12.8)(=)	25639.6 (11.7)(+)	24207.9 (14.8)(=)	24208.3 (13.0)(=)	24605.5 (14.2)(+)	24205.7 (8.2)
X-n524-k153	154654.4 (106.6)(=)	154638.4 (114.2)(=)	154745.2 (104.6)(+)	154622.5 (131.9)(=)	154605.2 (116.4)(=)	154809.6 (126.9)(+)	154607.9 (73.6)
P01	582.5 (11.6)(=)	580.2 (10.1)(=)	581.6 (2.9)(=)	581.9 (3.6)(=)	581.7 (3.5)(+)	584.1 (1.2)(+)	577.2 (8.0)
P02	488.2 (14.9)(+)	508.3 (17.8)(+)	489.0 (12.5)(+)	496.2 (4.8)(+)	482.1 (6.2)(+)	492.0 (14.0)(+)	473.8 (10.0)
P03	659.9 (2.1)(+)	655.9 (6.6)(+)	658.7 (4.8)(+)	651.2 (5.3)(+)	647.8 (4.8)(=)	643.6 (4.4)(=)	645.4 (4.3)
P04	1006.2 (7.7)(=)	1035.0 (21.8)(+)	1011.7 (15.5)(+)	1021.2 (11.6)(+)	1013.4 (15.0)(=)	1018.2 (8.2)(+)	1001.6 (9.9)
P05	764.8 (2.1)(+)	754.5 (7.5)(=)	754.4 (6.1)(=)	752.9 (4.3)(=)	763.8 (5.9)(+)	762.1 (5.5)(+)	752.0 (8.2)
P06	897.2 (10.1)(+)	883.5 (16.0)(=)	890.8 (5.6)(+)	875.6 (15.7)(=)	886.2 (18.2)(=)	890.5 (11.8)(+)	879.2 (14.9)
P07	894.8 (4.9)(=)	909.2 (12.5)(+)	886.5 (14.5)(=)	904.2 (20.0)(=)	894.6 (10.7)(=)	892.3 (5.3)(=)	890.3 (8.6)
P08	4441.9 (20.6)(=)	4436.8 (30.2)(=)	4456.2 (7.4)(=)	4414.0 (12.4)(-)	4455.1 (22.4)(=)	4401.4 (7.6)(-)	4449.5 (27.1)
P09	3909.1 (11.9)(=)	3910.9 (6.6)(=)	3909.9 (12.1)(=)	3904.9 (12.5)(=)	3902.1 (18.8)(=)	3905.9 (14.1)(=)	3906.3 (14.7)
P10	3676.5 (28.2)(=)	3685.5 (11.6)(=)	3676.9 (8.2)(=)	3687.8 (7.4)(=)	3683.4 (14.8)(=)	3673.5 (4.8)(-)	3681.0 (23.4)
P14_80_2	5652.0 (130.9)(=)	5630.6 (81.93)(=)	5603.7 (52.1)(=)	5630.9 (112.0)(=)	5599.9 (63.8)(=)	5591.6 (88.6)(=)	5639.9 (87.12)
P15_160_4	8921.8 (78.2)(=)	8946.4 (68.38)(=)	8918.2 (53.6)(=)	8884.9 (32.4)(=)	8902.5 (32.3)(=)	8890.0 (95.9)(=)	8932.7 (75.28)
P16_160_4	9348.6 (117.1)(+)	9358.8 (37.81)(+)	9342.0 (109.7)(+)	9334.4 (65.8)(+)	9296.6 (82.3)(=)	9355.4 (59.7)(+)	9287.4 (78.31)
P17_160_4	9932.2 (138.8)(=)	9986.6 (102.38)(=)	9893.6 (82.0)(=)	9953.9 (131.7)(=)	9866.5 (75.6)(=)	9901.3 (129.9)(=)	9863.2 (316.37)
P18_240_6	13411.9 (256.9)(+)	13355.8 (112.63)(+)	13332.5 (116.5)(+)	13296.9 (53.6)(+)	13371.9 (42.1)(+)	13277.5 (79.6)(=)	13226.5 (98.76)
P19_240_6	14316.4 (133.0)(+)	14155.6 (116.62)(=)	14272.8 (147.2)(=)	14045.5 (72.5)(=)	14283.3 (55.8)(+)	14070.8 (154.1)(=)	14158.3 (206.56)
P20_240_6	15199.6 (138.3)(+)	15115.3 (152.91)(+)	15193.4 (150.5)(+)	15094.0 (90.0)(=)	14966.3 (94.4)(=)	15142.9 (123.8)(=)	15058.8 (176.64)
P21_360_9	20178.7 (101.1)(+)	20067.5 (113.46)(+)	20120.6 (205.7)(+)	19985.9 (108.9)(+)	20036.8 (32.3)(+)	19993.6 (181.7)(+)	19881.1 (153.83)
P22_360_9	22447.1 (233.0)(+)	22450.3 (126.35)(+)	22532.1 (373.9)(+)	22329.8 (112.1)(=)	22340.6 (179.1)(+)	22415.0 (114.8)(+)	22174.6 (360.85)
P23_360_9	26406.9 (594.1)(+)	25952.2 (510.81)(=)	26059.8 (721.5)(+)	25823.9 (404.8)(=)	26061.0 (212.1)(+)	25766.3 (188.3)(=)	25735.0 (302.19)
W/T/L	17/13/0	15/15/0	18/12/0	9/20/1	13/17/0	14/13/3	/

The best results for each instance are highlighted in bold. The reported significance test is derived from all instances. The full results on all the instances can be found in the supplementary material.

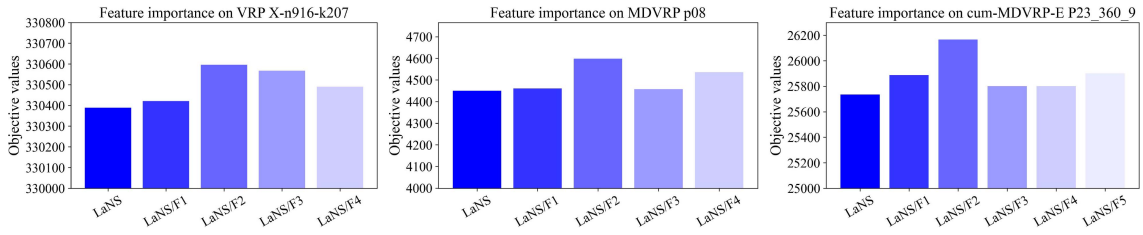


Fig. 4. The feature importance investigation. 0: All features are used. 1: Remove location feature. 2: Remove the adjacency feature. 3: Remove the demand feature. 4: Remove the capacity feature. 5: Remove battery feature.

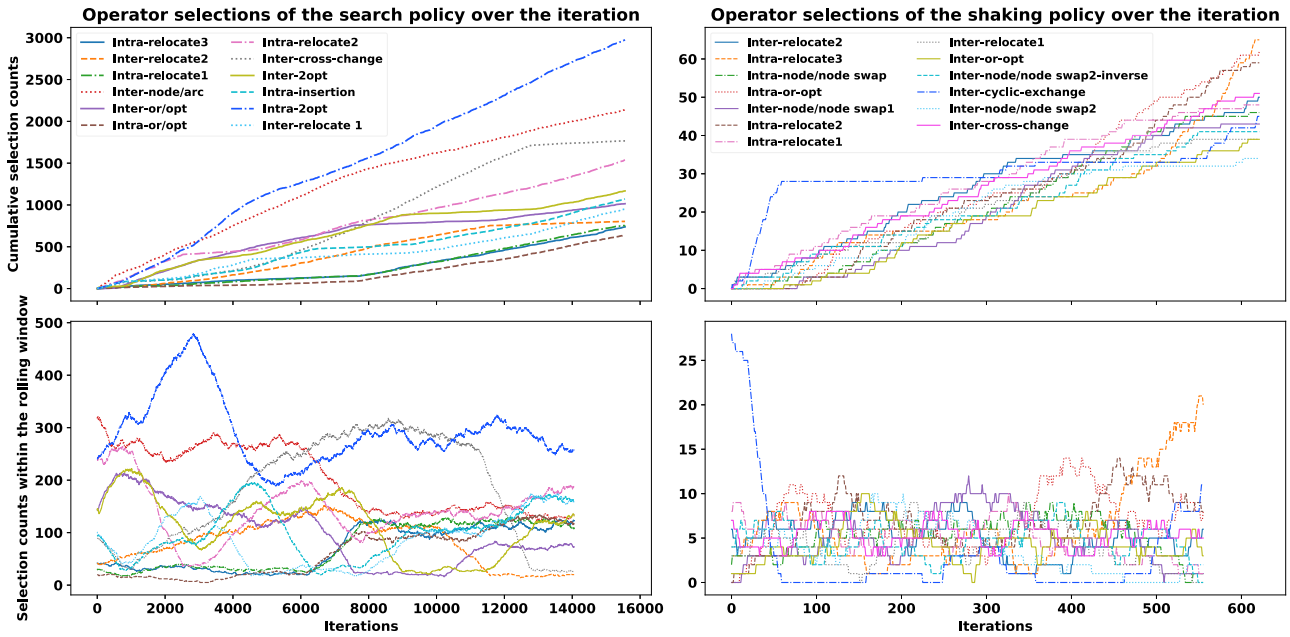


Fig. 5. The operator selection of the shaking/search policy over the iterations on the MDVRP instance of p08. For the search policy, the rolling window length equals 1000 while for the shaking policy equals 100.

its two counterparts, LaNS-F_{PPO} and LaNS-F_{DQN}, implying that replacing the proposed feature representations from LaNS leads to inferior performance. This suggests that the proposed feature representations are more informative for guiding the learning process. PPO and DQN utilize macro characteristics to construct problem features, overlooking potentially more informative features such as route structure, which limits their effectiveness in providing learning guidance knowledge.

In summary, it can be concluded that LaNS is better than the state-of-the-art AOS methods. Besides, the effectiveness of the MARL-based operator selection and the newly proposed feature representations has been validated.

C. Investigation of Feature Tensor

To gain a deep insight into the newly developed feature tensor, we remove the features one by one and observe the ultimate objective value using other features. LaNS/Fi denotes the i-th feature matrix in Section III-B is removed from the representations. The results on representative instances are shown in Fig. 4. Generally, the deep blue bar (LaNS) has the least objective value across all instances, demonstrating that all devised features contribute to enhancing guidance knowledge learning. Besides, it is observed that removing the adjacency feature may lead to significant degeneration. The reason is that the adjacency feature is extracted from the current solution, which contains crucial spatial information about the route structure and reflects the problem state. On the other hand, instance-related features (i.e., the location feature) are likely to have less impact on learning as the change of ultimate objective values is smaller after removing them.

D. Analysis of Operator Usages

As previously noted, online RL agents adaptively select suitable operators based on the search state. We analyzed the cumulative counts and per-unit time selections of operators chosen by the shaking and search policies during iteration, as shown in Fig. 5. The selection frequency of each operator dynamically shifts during optimization. Initially, the search policy favors operators with larger perturbations, such as Inter-node/arc and Inter-cross-change, to explore extensively. However, as the search progresses, it opts for operators with smaller magnitudes like Intra-relocate2, Intra-insertion, and Intra-2opt to avoid rapid solution degradation. Conversely, the shaking policy initially prefers operators that switch routes between depots, aiding in

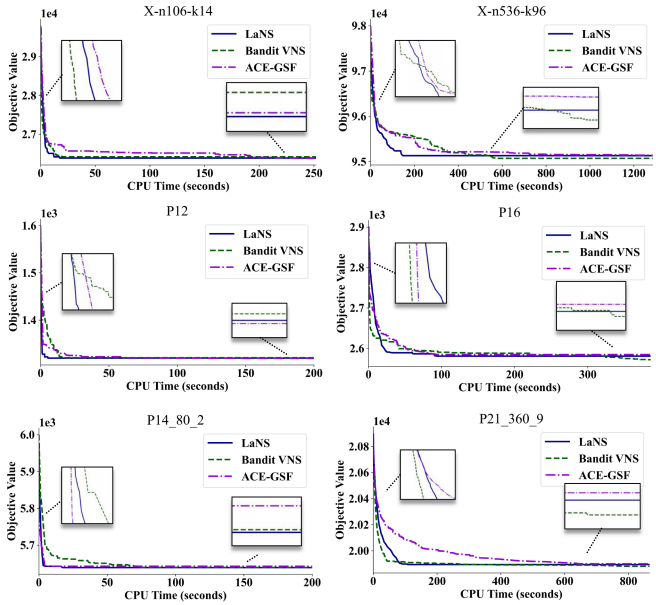


Fig. 6. Convergence curves of LaNS and different reward shaping methods on representative instances.

matching the customers to suitable depots (e.g., Inter-cyclic-exchange, Inter-cross-change), and later favors operators like Intra-relocate3 and Intra-or-opt that significantly destroy the route structure in order to escape local optima.

E. Analysis of Reward Diminishing Issue

In the proposed LaNS algorithm, the rewards for training the policies are proportional to the improvement ratio, like most existing learning-based AOS methods. However, as optimization progresses towards stability, the improvement ratio diminishes, leading to reduced rewards collected over time. This non-stationary issue might hinder the RL agent's ability to explore effectively within its policy space. More recent works have noticed this issue and proposed reward shaping techniques to alleviate this problem. The major idea is to increase the scale of reward values as the optimization progresses into later stages. To analyze the impact of reward diminishing issue on the performance, we adopt the reward shaping methods in Bandit VNS [20] and ACE-GSF [69] to the proposed LaNS algorithm. The convergence curve over time on representative instances is shown in Fig. 6. We can observe that the compared algorithms have similar convergence speed and convergence quality on small-scale instances. In the case of large-scale instances, Bandit VNS continues to learn the policy during the later stages of optimization and achieves better objective value on the selected instances. Conversely, ACE-GSF performs even worse than LaNS. The explanation is that Bandit VNS employs a concept drift detection scheme to automatically adjust reward values, whereas ACE-GSF uses a pre-defined stage indicator, whose parameters are sensitive to instance characteristics and require sophisticated tuning. The comparison results suggest that employing reward shaping to mitigate the reward issue can enhance policy learning among RL agents in the later stages of optimization. However, both the module and its parameters require careful design. These findings motivate us to develop more advanced techniques to address the diminishing reward issue and further enhance performance in our future work.

VI. Conclusions and Future Work

In this paper, we develop a Learning-aided Neighborhood Search algorithm (LaNS) for Vehicle Routing Problems (VRP). This learning-aided framework facilitates automated algorithm design by integrating traditional heuristic methods and machine learning, which can avoid the laborious workload of domain experts in designing the operator sequence. Moreover, a new informative feature tensor is constructed to provide comprehensive information for the agents to learn guidance knowledge. Extensive experiments conducted across a wide range of VRP benchmarks with different scales demonstrate that LaNS can have a better performance than the state-of-the-art neighborhood search algorithms and the adaptive operator selection methods. Further analysis reveals that the collaborative efforts of shaking and search policies significantly enhance the efficiency of the optimization process, steering it towards (near) optimal solutions and adaptively shaping the distribution of operator selection. Additionally, it has been observed that all the formulated features play a crucial role in augmenting learning performance. Notably, the adjacency feature exerts a more substantial impact, underscoring its importance.

In the future, we plan to delve into the automatic extraction of feature representations using graph neural networks. These networks have proven effective in capturing non-euclidean data, such as the route structures inherent in VRP. The development of novel neural network architectures and feature representation techniques holds considerable promise and warrants further exploration.

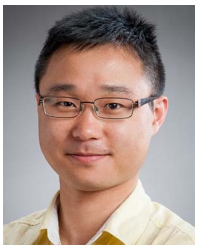
References

[1] E. Uchoa, D. Pecin, A. Pessoa, M. Poggi, T. Vidal, and A. Subramanian, "New benchmark instances for the capacitated vehicle routing problem," *Eur. J. Oper. Res.*, vol. 257, no. 3, pp. 845–858, 2017.
[2] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Manage. Sci.*, vol. 6, no. 1, pp. 80–91, 1959.
[3] L. Costa, C. Contardo, and G. Desaulniers, "Exact branch-price-and-cut algorithms for vehicle routing," *Transp. Sci.*, vol. 53, no. 4, pp. 946–985, 2019.
[4] R. Baldacci, A. Mingozzi, and R. Roberti, "Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints," *Eur. J. Oper. Res.*, vol. 218, no. 1, pp. 1–6, 2012.
[5] J. M. Weinand et al., "Research trends in combinatorial optimization," *Int. Trans. Oper. Res.*, vol. 29, no. 2, pp. 667–705, 2022.
[6] R. Elshaer and H. Awad, "A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants," *Comput. Ind. Eng.*, vol. 140, 2020, Art. no. 106242.
[7] D. P. Cuervo, P. Goos, K. Sörensen, and E. Arráiz, "An iterated local search algorithm for the vehicle routing problem with backhauls," *Eur. J. Oper. Res.*, vol. 237, no. 2, pp. 454–464, 2014.
[8] X. Wang, T.-M. Choi, Z. Li, and S. Shao, "An effective local search algorithm for the multidepot cumulative capacitated vehicle routing problem," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 50, no. 12, pp. 4948–4958, Dec. 2020.
[9] D. Pisinger and S. Ropke, "A general heuristic for vehicle routing problems," *Comput. Oper. Res.*, vol. 34, no. 8, pp. 2403–2435, 2007.
[10] M. Sadati, D. Aksen, and N. Aras, "A trilevel *r*-interdiction selective multi-depot vehicle routing problem with depot protection," *Comput. Oper. Res.*, vol. 123, 2020, Art. no. 104996.
[11] R. Liu and Z. Jiang, "A hybrid large-neighborhood search algorithm for the cumulative capacitated vehicle routing problem with time-window constraints," *Appl. Soft Comput.*, vol. 80, pp. 18–30, 2019.

[12] S. Nunes Bezerra, M. J. F. Souza, S. R. de Souza, and V. Nazário Coelho, "A VNS-based algorithm with adaptive local search for solving the multi-depot vehicle routing problem," in *Proc. 6th Int. Conf. Variable Neighborhood Search*, Springer, 2018, pp. 167–181.
[13] X. Xu, J. Li, M. Zhou, and X. Yu, "Precedence-constrained colored traveling salesman problem: An augmented variable neighborhood search approach," *IEEE Trans. Cybern.*, vol. 52, no. 9, pp. 9797–9808, Sep. 2022.
[14] D. Lei, M. Li, and L. Wang, "A two-phase meta-heuristic for multiobjective flexible job shop scheduling problem with total energy consumption threshold," *IEEE Trans. Cybern.*, vol. 49, no. 3, pp. 1097–1109, Mar. 2019.
[15] X. Li, L. Gao, Q. Pan, L. Wan, and K.-M. Chao, "An effective hybrid genetic algorithm and variable neighborhood search for integrated process planning and scheduling in a packaging machine workshop," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 49, no. 10, pp. 1933–1945, Oct. 2019.
[16] J. Pei, Y. Mei, J. Liu, and X. Yao, "An investigation of adaptive operator selection in solving complex vehicle routing problem," in *Proc. Pacific Rim Int. Conf. Artif. Intell.*, Springer, 2022, pp. 562–573.
[17] Q. Lin et al., "Adaptive composite operator selection and parameter control for multiobjective evolutionary algorithm," *Inf. Sci.*, vol. 339, pp. 332–352, 2016.
[18] H. Lu, X. Zhang, and S. Yang, "A learning-based iterative method for solving vehicle routing problems," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–15. [Online]. Available: https://openreview.net/forum?id=BJe1334YDH
[19] Y. Tian, X. Li, H. Ma, X. Zhang, K. C. Tan, and Y. Jin, "Deep reinforcement learning based adaptive operator selection for evolutionary multi-objective optimization," *IEEE Trans. Emerg. Top. Comput.*, vol. 7, no. 4, pp. 1051–1064, Aug. 2023.
[20] P. Kalatzantonakis, A. Sifaleras, and N. Samaras, "A reinforcement learning-variable neighborhood search method for the capacitated vehicle routing problem," *Expert Syst. Appl.*, vol. 213, 2023, Art. no. 118812.
[21] R. Durgut, M. E. Aydin, and I. Atli, "Adaptive operator selection with reinforcement learning," *Inf. Sci.*, vol. 581, pp. 773–790, 2021.
[22] S. S. Choong, L.-P. Wong, and C. P. Lim, "Automatic design of hyper-heuristic based on reinforcement learning," *Inf. Sci.*, vol. 436, pp. 89–107, 2018.
[23] W. Yi, R. Qu, L. Jiao, and B. Niu, "Automated design of metaheuristics using reinforcement learning within a novel general search framework," *IEEE Trans. Evol. Comput.*, vol. 27, no. 4, pp. 1072–1084, Aug. 2023.
[24] R. Qi, J.-Q. Li, J. Wang, H. Jin, and Y.-Y. Han, "QMOEA: A Q-learning-based multiobjective evolutionary algorithm for solving time-dependent green vehicle routing problems with time windows," *Inf. Sci.*, vol. 608, pp. 178–201, 2022.
[25] N. R. Sabar, M. Ayob, G. Kendall, and R. Qu, "A dynamic multiarmed bandit-gene expression programming hyper-heuristic for combinatorial optimization problems," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 217–228, Feb. 2015.
[26] O. Bräysy and M. Gendreau, "Vehicle routing problem with time windows, part II: Metaheuristics," *Transp. Sci.*, vol. 39, no. 1, pp. 119–139, 2005.
[27] J. Renaud, G. Laporte, and F. F. Boctor, "A tabu search heuristic for the multi-depot vehicle routing problem," *Comput. Oper. Res.*, vol. 23, no. 3, pp. 229–235, 1996.
[28] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, "Vehicle routing problems for drone delivery," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 47, no. 1, pp. 70–85, Jan. 2017.
[29] K. Corona-Gutiérrez, S. Nucamendi-Guillén, and E. Lalla-Ruiz, "Vehicle routing with cumulative objectives: A state of the art and analysis," *Comput. Ind. Eng.*, vol. 169, 2022, Art. no. 108054.
[30] S. Erdoğan and E. Miller-Hooks, "A green vehicle routing problem," *Transport. Res. E.*, vol. 48, no. 1, pp. 100–114, 2012.
[31] M. Mahmoudi and X. Zhou, "Finding optimal solutions for vehicle routing problem with pickup and delivery services with time windows: A dynamic programming approach based on state–space–time network representations," *Transport. Res. B-Meth*, vol. 89, pp. 19–42, 2016.
[32] C. Archetti, N. Bianchessi, and M. G. Speranza, "Branch-and-cut algorithms for the split delivery vehicle routing problem," *Eur. J. Oper. Res.*, vol. 238, no. 3, pp. 685–698, 2014.
[33] M. Jin, K. Liu, and B. Eksioglu, "A column generation approach for the split delivery vehicle routing problem," *Oper. Res. Lett.*, vol. 36, no. 2, pp. 265–270, 2008.
[34] N. A. Kyriakakis, I. Sevastopoulos, M. Marinaki, and Y. Marinakis, "A hybrid tabu search–variable neighborhood descent algorithm for the cumulative capacitated vehicle routing problem with time windows in humanitarian applications," *Comput. Ind. Eng.*, vol. 164, 2022, Art. no. 107868.
[35] Y. Wu, F. Pan, S. Li, Z. Chen, and M. Dong, "Peer-induced fairness capacitated vehicle routing scheduling using a hybrid optimization aco–vns algorithm," *Soft Comput.*, vol. 24, pp. 2201–2213, 2020.
[36] N. A. Kyriakakis, M. Marinaki, N. Matsatsinis, and Y. Marinakis, "A cumulative unmanned aerial vehicle routing problem approach for humanitarian coverage path planning," *Eur. J. Oper. Res.*, vol. 300, no. 3, pp. 992–1004, 2022.
[37] F. Arnold and K. Sörensen, "Knowledge-guided local search for the vehicle routing problem," *Comput. Oper. Res.*, vol. 105, pp. 32–46, 2019.
[38] J. G. C. Costa, Y. Mei, and M. Zhang, "Guided local search with an adaptive neighbourhood size heuristic for large scale vehicle routing problems," in *Proc. ACM Genet. Evol. Comput. Conf.*, 2022, pp. 213–221.
[39] Q. Zhang and S. Xiong, "Routing optimization of emergency grain distribution vehicles using the immune ant colony optimization algorithm," *Appl. Soft Comput.*, vol. 71, pp. 917–925, 2018.
[40] F. Wan, H. Guo, J. Li, M. Gu, W. Pan, and Y. Ying, "A scheduling and planning method for geological disasters," *Appl. Soft Comput.*, vol. 111, 2021, Art. no. 107712.
[41] X. Wang, T.-M. Choi, H. Liu, and X. Yue, "A novel hybrid ant colony optimization algorithm for emergency transportation problems during post-disaster scenarios," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 48, no. 4, pp. 545–556, Apr. 2018.
[42] S. U. Ngueveu, C. Prins, and R. W. Calvo, "An effective memetic algorithm for the cumulative capacitated vehicle routing problem," *Comput. Oper. Res.*, vol. 37, no. 11, pp. 1877–1885, 2010.
[43] R. G. Ribeiro, L. P. Cota, T. A. Euzébio, J. A. Ramírez, and F. G. Guimarães, "Unmanned-aerial-vehicle routing problem with mobile charging stations for assisting search and rescue missions in postdisaster scenarios," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 52, no. 11, pp. 6682–6696, Nov. 2022.
[44] T. Vidal, "Hybrid genetic search for the CVRP: Open-source implementation and swap* neighborhood," *Comput. Oper. Res.*, vol. 140, 2022, Art. no. 105643.
[45] P. Garrido and C. Castro, "Stable solving of CVRPs using hyperheuristics," in *Proc. 11th Annu. Conf. Companion Genet. Evol. Comput.*, 2009, pp. 255–262.
[46] P. Garrido, C. Castro, and É. Monfroy, "Towards a flexible and adaptable hyperheuristic approach for VRPs," in *Proc. Int. Conf. Artif. Intell.*, 2009, pp. 311–317.
[47] J. H. Drake, N. Kililis, and E. Özcan, "Generation of VNS components with grammatical evolution for vehicle routing," in *Proc. Eur. Conf. Genet. Prog.*, Springer, 2013, pp. 25–36.
[48] J. Mlejnek and J. Kubalik, "Evolutionary hyperheuristic for capacitated vehicle routing problem," in *Proc. 15th Annu. Conf. Companion Genet. Evol. Comput.*, 2013, pp. 219–220.
[49] J. G. C. Costa, Y. Mei, and M. Zhang, "Learning initialisation heuristic for large scale vehicle routing problem with genetic programming," in *Proc. IEEE Congr. Evol. Comput.*, 2021, pp. 1864–1871.
[50] J. Jacobsen-Grocott, Y. Mei, G. Chen, and M. Zhang, "Evolving heuristics for dynamic vehicle routing with time windows using genetic programming," in *Proc. IEEE Congr. Evol. Comput.*, 2017, pp. 1948–1955.
[51] G. Gao, Y. Mei, B. Xin, Y.-H. Jia, and W. N. Browne, "Automated coordination strategy design using genetic programming for dynamic multipoint dynamic aggregation," *IEEE Trans. Cybern.*, vol. 52, no. 12, pp. 13 521–13 535, Dec. 2022.
[52] N. R. Sabar and G. Kendall, "Population based Monte Carlo tree search hyper-heuristic for combinatorial optimization problems," *Inf. Sci.*, vol. 314, pp. 225–239, 2015.
[53] N. Mladenović and P. Hansen, "Variable neighborhood search," *Comput. Oper. Res.*, vol. 24, no. 11, pp. 1097–1100, 1997.
[54] H. Xu, W. Fan, T. Wei, and L. Yu, "An or-opt NSGA-II algorithm for multi-objective vehicle routing problem with time windows," in *Proc. 2008 IEEE Int. Conf. Automat. Sci. Eng.*, 2008, pp. 309–314.
[55] M. Gendreau, A. Hertz, and G. Laporte, "New insertion and postoptimization procedures for the traveling salesman problem," *Oper. Res.*, vol. 40, no. 6, pp. 1086–1094, 1992.
[56] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[57] C. S. de Witt et al., "Is independent learning all you need in the starcraft multi-agent challenge?," 2020, *arXiv: 2011.09533*.
[58] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.
[59] İ. Kara, B. Y. Kara, and M. K. Yetiş, "Cumulative vehicle routing problems," in *Veh. Routing Problem*, vol. 6, pp. 85–98, 2008.

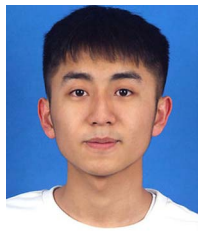
[60] M. E. H. Sadati, B. Çatay, and D. Aksen, "An efficient variable neighborhood search with tabu shaking for a class of multi-depot vehicle routing problems," *Comput. Oper. Res.*, vol. 133, 2021, Art. no. 105269.
[61] T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei, "A hybrid genetic algorithm for multidepot and periodic vehicle routing problems," *Oper. Res.*, vol. 60, no. 3, pp. 611–624, 2012.
[62] T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins, "A unified solution framework for multi-attribute vehicle routing problems," *Eur. J. Oper. Res.*, vol. 234, no. 3, pp. 658–673, 2014.
[63] F. B. de Oliveira, R. Enayatifar, H. J. Sadaei, F. G. Guimaraes, and J.-Y. Potvin, "A cooperative coevolutionary algorithm for the multi-depot vehicle routing problem," *Expert Syst. Appl.*, vol. 43, pp. 117–130, 2016.
[64] E. Lalla-Ruiz and S. Voss, "POPMUSIC as a matheuristic for the berth allocation problem," *Ann. Math. Artif. Intell.*, vol. 76, pp. 173–189, 2016.
[65] E. Lalla-Ruiz, S. Voß, C. Expósito-Izquierdo, B. Melián-Batista, and J. M. Moreno-Vega, "A popmusic-based approach for the berth allocation problem under time-dependent limitations," *Ann. Oper. Res.*, vol. 253, pp. 871–897, 2017.
[66] E. Lalla-Ruiz and S. Vo, "A popmusic approach for the multi-depot cumulative capacitated vehicle routing problem," *Optim. Lett.*, vol. 14, no. 3, pp. 671–691, 2020.
[67] M. Niu, R. Liu, and H. Wang, "A max-min ant system based on decomposition for the multi-depot cumulative capacitated vehicle routing problem," in *Proc. 2021 IEEE Congr. Evol. Comput.*, 2021, pp. 620–627.
[68] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Cáceres, M. Birattari, and T. Stützle, "The irace package: Iterated racing for automatic algorithm configuration," *Oper. Res. Perspect.*, vol. 3, pp. 43–58, 2016.
[69] W. Yi and R. Qu, "Automated design of search algorithms based on reinforcement learning," *Inf. Sci.*, vol. 649, 2023, Art. no. 119639.

Tong Guo (Graduate Student Member, IEEE) received the BS degree from Shen Yuan Honors College, Beihang University, in 2019, where he is currently working toward the PhD degree with the Department of Electronic and Information Engineering. He is also a visiting student with the School of Engineering and Computer Science, Victoria University of Wellington. His research interests include evolutionary computation and reinforcement learning.



Yi Mei (Senior Member, IEEE) is currently an associate professor with the School of Engineering and Computer Science and associate dean (Research) with the Faculty of Engineering, Victoria University of Wellington, New Zealand. His research interests include evolutionary computation and machine learning for combinatorial optimisation, genetic programming, hyper-heuristics, and explainable AI. He is the founding chair of *IEEE Taskforce on Evolutionary Scheduling and Combinatorial Optimisation*. He serves as a vice-chair of the IEEE CIS Emergent Technologies Technical Committee, and a member of Intelligent Systems Applications Technical Committee. He is a fellow of Engineering New Zealand.

Mengjie Zhang (Fellow, IEEE) is currently a professor of Computer Science, the head of the Evolutionary Computation and Machine Learning Research Group, Victoria University of Wellington, New Zealand. His current research interests include genetic programming, image analysis, feature selection and reduction, job-shop scheduling, and evolutionary deep learning and transfer learning. He has published more than 800 research papers in refereed international journals and conferences. He is a fellow of the Royal Society of New Zealand, a fellow of Engineering New Zealand, and an IEEE distinguished lecturer.



Haoran Zhao received the BS from the School of Computer Science and Technology, University of Electronic Science and Technology of China, Sichuan, China, in 2022. He is currently working toward the MS degree with the School of Electronic and Information Engineering, Beihang University, Beijing 100191, China. His current research interests include reinforcement learning and machine learning.

Kaiquan Cai received the PhD degree from Beihang University, Beijing, China, in 2013, where he is currently a professor with the School of Electronic and Information Engineering and the deputy director of the National Key Laboratory of Communications, Navigation, and Surveillance Systems for Air Traffic Management. His research interests include intelligent air navigation and networked collaborative air traffic management.

Wenbo Du (Member, IEEE) received the BS and PhD degrees from the School of Computer Science and Technology, University of Science and Technology of China, Hefei, China, in 2005 and 2010, respectively. He is a professor with the School of Electronic and Information Engineering, Beihang University, Beijing, China. His current research interests include data science and intelligent transportation.