Machine Learning to Solve Vehicle Routing Problems: A Survey

Aigerim Bogyrbayeva, Meraryslan Meraliyev, Taukekhan Mustakhov, and Bissenbay Dauletbayev

***Abstract*— This paper provides a systematic overview of machine learning methods applied to solve NP-hard Vehicle Routing Problems (VRPs). Recently, there has been great interest from both the machine learning and operations research communities in solving VRPs either through pure learning methods or by combining them with traditional handcrafted heuristics. We present a taxonomy of studies on learning paradigms, solution structures, underlying models, and algorithms. Detailed results of state-of-the-art methods are presented, demonstrating their competitiveness with traditional approaches. The survey highlights the advantages of the machine learning-based models that aim to exploit the symmetry of VRP solutions. The paper outlines future research directions to incorporate learning-based solutions to address the challenges of modern transportation systems.**

***Index Terms*— Reinforcement learning, supervised learning, neural combinatorial optimization, vehicle routing.**

I. Introduction

COST-EFFECTIVE logistics systems define the competitiveness of companies, and the relation of logistics expenditure to GDP indicates the effectiveness of business operations in a country. For instance, in 2018, USA businesses spent 10.4% of their revenue on transportation costs alone, while the overall logistics expenditure constituted 8% of GDP [1]. Along with increased fuel prices, transportation costs are mainly influenced by last-mile delivery, defined as transporting goods from a warehouse to a customer's location. With the increased demand for online sales, the last-mile delivery system's effectiveness has become essential as it comprises 50% of the total transportation costs [2]. Also, the carbon dioxide footprint is another emerging concern in logistics systems. To overcome the above-outlined challenges, efficiently solving vehicle routing problems has been of great interest to both practitioners and researchers.

The Vehicle Routing Problem, in general, is defined on either a directed or undirected graph $\mathbb{G}(\mathbb{V}, \mathbb{E})$ consisting of a set of nodes $\mathbb{V} = \{v_0, \ldots, v_n\}$ and a set of edges or arcs. We focus on an undirected graph with a set of edges $\mathbb{E} = \{(v_i, v_j) : i < j, v_i, v_j \in \mathbb{V}\}$. VRP aims to construct routes with minimal total cost for M identical vehicles leaving from a depot, v_0, to visit all nodes,$\mathbb{V} \setminus v_0$, representing customers with non-negative demand d_v, and return to the depot, where each customer is visited only once. Different constraints are added to VRP to reflect the realistic routing challenges faced by delivery companies [3]. In the Capacitated Vehicle Routing Problem (CVRP), each vehicle is subjected to a maximum capacity Q_m such that the total demand of visited customers in its route does not exceed Q_m. In Vehicle Routing Problems with Time Windows (VRPTW), vehicles must arrive at the customer location within specified time windows. However, all the above variations of the VRP belong to vertex routing problems. Some applications, such as mail delivery and bus routing, are concerned with visiting arcs called arc routing problems (ARPs). The key difference between vertex and arc routing problems is that the former involves finding a path between a set of vertices, while the latter involves finding a path between a set of arcs or edges.

As generalizations of the Traveling Salesman Problem (TSP), VRPs belong to the family of Combinatorial Optimization (CO) problems and are known to be NP-hard [4]. Therefore, approximation or heuristic algorithms have been developed to solve large-size VRPs [5], [6], [7], while Mixed Integer Programming (MIP) is used to find exact solutions on small instances. The handcrafted heuristics are built on the experts' domain knowledge about the specifics of the problem's structure. Even though such heuristics often produce solutions in a short amount of time, they fail to generalize to slight changes in the inputs and must be solved from scratch [8].

On the other hand, machine learning methods have shown great success across many applications due to advancements in algorithms and hardware. Machine Learning, in general, can be viewed as applied statistics, where the computational power of computers is used to *learn* approximate functions instead of explicitly writing computer programs. The main advantage of machine learning models is their ability to generalize to input problems coming from identical distributions and produce solutions instantaneously [9]. This can be invaluable in practice to solve online routing problems with dynamically changing inputs. Therefore, there has been a surge of studies in recent years aiming to develop novel models and training algorithms for routing problems using machine learning methods to achieve near-optimal solutions.

Manuscript received 10 October 2022; revised 23 May 2023 and 8 September 2023; accepted 8 November 2023. This work was supported by the funding for Young Scientists' Scientific and Technical Projects for 2023–2025, provided by the Ministry of Science and Higher Education of the Republic of Kazakhstan under Grant IRN AP19575607. The Associate Editor for this article was M. A. Khan. *(Corresponding author: Meraryslan Meraliyev.)*

The authors are with the Department of Computer Science, SDU University, 040900 Kaskelen, Kazakhstan (e-mail: aigerim.bogyrbayeva@sdu.edu.kz; meraryslan.meraliyev@sdu.edu.kz; taukekhan.mustakhov@sdu.edu.kz; b.dauletbayev@sdu.edu.kz).

Digital Object Identifier 10.1109/TITS.2023.3334976

There are two main learning paradigms used for solving VRPs. In supervised learning (SL), a model is trained on a training dataset consisting of input problems and corresponding solutions. In Reinforcement Learning (RL), VRPs are sequential decision-making problems and rely on the Markov Decision Process (MDP) to construct solutions. However, pure learning-based heuristics have disadvantages such as generalization, poor solution quality, and scalability, leading to new methods that combine traditional heuristics with learning methods. An increasingly growing literature has attracted the attention of both machine learning and operations research communities [10]. Therefore, this survey aims to systematically present the recent advancements in learning methods to solve routing problems and discuss the challenges and opportunities for future research. We believe the survey will benefit researchers interested in solving VRPs either using pure learning methods or hybrid methods that combine learning and construction heuristics.

There are a few studies that focus on machine learning methods used to solve VRPs. For instance, [10] presents general methodological frameworks to merge machine learning and optimization methods to solve CO problems, omitting problem-specific approaches for VRPs. Similarly, [11] and [12] focus on only Deep Reinforcement Learning (DRL) approaches to tackle a set of well-known CO problems, while [13] and [14] discuss machine learning approaches to solve CO problems in networking and manufacturing, respectively. Reference [15] limit their survey to Graph Neural Networks applications to solve CO problems, and [16] focuses only on TSP. Reference [17]vprovides an overview of data analytics and machine learning approaches used for modeling and optimization of VRPs. With the heavy focus on hybrid methods,the paper does not provide any computational studies to compare recently proposed models and demonstrate state-of-the-art results. Although [18] incorporates experimental analysis, it falls short of encompassing the recent studies. In contrast, this study includes a comprehensive computational study, including a large number of models, their variants, and different problem types using well-known datasets. Also, we provide the detailed modeling of single and multiple VRPs used in many realistic VRP variants that have also been missed in [18]. Nevertheless, we consider our work to be complementary to the existing studies.

In total, our contributions can be summarized as follows: i) this survey comprehensively summarizes the most recent machine learning approaches to solve various VRPs, including both pure end-to-end learning and hybrid methods; ii) we present the computational studies to compare the surveyed paper results on random and well-known benchmark instances to demonstrate the state-of-the-art.

The survey indicates the benefits of learning-based models to solve medium-sized graphs consisting of 100 nodes, to produce comparable solutions with the optimization methods with less computational time. Within learning-based models, both pure end-to-end and hybrid models are comparable in terms of solution quality. In both methods, the studies focused on the symmetry of VRP solutions perform better as compared to the studies that ignore such properties. However, the majority of the studies focus on routing a single vehicle with fully known inputs. This is in contrast with real-world applications of VRPs that often involve routing a fleet of vehicles with dynamic inputs changing over time, indicating challenging yet important future research directions.

We have the following outline for the paper: in Section II, we present the central taxonomy and discuss end-to-end learning methods; in Section III, we focus on hybrid methods and present different solution structures; in Section IV we present single and multiple VRP formulations; in Section V, we present our experimental results; in Section VI, we discuss the future research directions; in Section VII, we finalize with the concluding remarks.

II. Taxonomy of Learning Methods for Routing Problems

It is challenging to classify the wide range of learning methods used to solve routing problems using a single classification. However, one useful classification is based on the structure of the solutions, which distinguishes between solutions that use only learning methods and those that combine learning methods with existing non-learning methods. In *end-to-end* learning methods, either SL or RL is used to solve a problem from the beginning until the final solution. In *hybrid* methods, learning methods are used either as a primary method to construct feasible and efficient solutions, which will later be further improved with construction heuristics, or learning methods facilitate solving the inner problems of the existing non-learning methods to solve routing problems.

One way to classify routing problems is based on the number of vehicles involved, either single-vehicle or multi-vehicle. Single-vehicle routing problems (SVRPs) involve finding the optimal route for a single vehicle to visit a set of locations, while multi-vehicle problems require addressing the challenge of routing multiple vehicles. A central controller is commonly employed to manage a fleet of heterogeneous or homogeneous vehicles. Since most studies focus on routing a single vehicle, the underlying problems are commonly referred to as TSP, CVRP, and VRPTW, while mTSP, mCVRP, and mVRPTW are used to indicate the presence of multiple vehicles in corresponding problems.

A. End-to-End SL for VRPs

In end-to-end models, purely learning methods construct a solution without any assistance from non-learning methods. *SL* requires having high-quality VRP solutions as labels that guide a model to find efficient solutions. Furthermore, depending on how the solutions are produced, we can divide them into *autoregressive* (AR) and *non-autoregressive* (NAR) categories. In AR SL methods, the routes are constructed step by step, one node at a time, while in NAR methods, the routes are constructed in a zero-shot manner.

In the case of TSP, to produce an NAR solution, the Held-Kalp [19] algorithm or well-known solvers such as Concorde [20] and LKH3 [21] are used to generate optimal solutions. Thus, for a given graph $\mathbb{G}(\mathbb{V}, \mathbb{E})$ in the two-dimensional Euclidean space, the output of the solver, π^*, is decomposed

TABLE I
THE SUMMARY OF END-TO-END SL METHODS WITH AUTOREGRESSIVE (AR) AND NONE-AUTOREGRESSIVE (NAR) SOLUTIONS

Name	Problems	AR	NAR	Loss	Label
Joshi et al. [23]	TSP		✓	Cross-entropy	Concorde
Milan et al. [25]	TSP	✓		Log-likelihood	Nearest Neighbours
Prates et al. [22]	a decision TSP			Binary cross-entropy	Concorde
Joshi et al. [26]	TSP	✓		Cross-entropy	Concorde
Joshi et al. [27]	TSP	✓	✓	Cross-entropy	Concorde
Van Knippenberg et al. [24]	CVRP, TSP	✓		Cross-entropy	Held-Kalp algorithm, LKH-3, Concorde

into the adjacency matrix, where for each $e_{i,j} \in \mathbb{E}$ we can compute the probability $p_{i,j}$:

$$p_{i,j} = \begin{cases} 1, & \text{if } e_{i,j} \text{ is in } \pi^* \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Following that, a deep learning model with a set of parameters θ, given the graph, is trained to produce an adjacency matrix with probabilities $\hat{p}_{i,j} \quad \forall e_{i,j} \in \mathbb{E}$. The loss is then defined as a binary cross-entropy:

$$\mathcal{L}(\theta) = p_{i,j} \log \hat{p}_{i,j} - (1 - p_{i,j}) \log(1 - \hat{p}_{i,j}) \tag{2}$$

The adjacency matrix produced by the model can either follow a *greedy search* by picking edges with the most significant probabilities or be coupled with *beam search* by preserving some number of candidate tours to be selected as the final solution.

On the other hand, an exact or high-quality approximation solution $\pi^* = [v_0, \ldots, v_n]$ is fed to a deep learning model to produce partial tours to maximize the conditional probability given below in AR SL methods:

$$\theta = \text{argmax} \log p(\pi|\pi^*, \theta) \tag{3}$$

$$p(\pi|\pi^*, \theta) = \prod_{i=1}^{n} p(v_i|v_0, \ldots, v_{i-1}, \pi^*, \theta). \tag{4}$$

Both methods utilize advanced deep learning models such as Graph Neural Networks (GNNs) [22] and Graph Convolution Networks [23] to extract features of a graph and deploy Memory Augmented Neural Networks [24] and Recurrent Neural Networks (RNNs) [25] to pass sequential information. Both methods require training separate sets of parameters for different graph sizes to produce near-optimal solutions for TSP [26], [27]. Table I summarizes studies focused on end-to-end SL.

B. End-to-End DRL for VRPs

While *tabular* RL methods are mostly used in the hybrid setting, in end-to-end methods, DRL is a popular choice due to the combinatorial nature of the action space of VRPs. We can categorize deep RL methods applied to VRPs as value-based and policy-based.

In value-based methods, such as Deep Q-learning for VRPs, neural networks are used to approximate the Q-function for each state and action pair, from which an optimal policy is derived. However, instead of using one-step updates, n-step TD is used as a target to consider delayed rewards. Further, *experience replay* is used to update Q-values with a batch of samples of the tuple $\{S_t, A_t, R_{t,t+n}, S_{t+n}\}$, where $R_{t,t+n} = \sum_{k=0}^{n} r_{t+k}$:

$$\hat{Q}_\theta(S_t, A_t) \leftarrow \hat{Q}_\theta(S_t, A_t) + \alpha[R_{t,t+n} + \gamma \max_a \hat{Q}_\theta \times (S_{t+n}, a_{t+n}) - \hat{Q}_\theta(S_t, A_t)]. \tag{5}$$

Experience replay is a technique used in RL, where a sample of the agent's experiences is stored and replayed during the learning process. Reference [28] points out that sampling a batch of stored experiences to update the neural network parameters results in fast convergence to solve combinatorial optimization problems, which is a general case when neural networks are used as approximation functions [29], [30].

Even though value-based models are generally sample-efficient as compared to policy-gradient methods, the routing policies are complex. Therefore, in most studies, VRP policies are directly learned through policy-gradient methods. In particular, a parametrized policy π_θ is approximated using the conditional probabilities and chain rule:

$$\pi_\theta(a|s) = \prod_{t=0}^{T} p(a_{t+1}|s_t, Y_t) \tag{6}$$

where Y_t is a set of visited nodes. In practice, the actor-critic structure is often used to represent two sets of neural networks to learn π_θ and $V_\theta(S_0)$ respectively, where $V_\theta(S_0)$ serves as a baseline. Mean Square Error is used to update the parameters of the critic θ_c with batch size B using total reward R:

$$\theta_c \leftarrow \theta_c + \alpha[\frac{1}{B} \sum_{j=1}^{B} \nabla_{\theta_c} (R^j - \hat{V}^j_{\theta_c}(S^j_0))^2] \tag{7}$$

Accordingly, the actor parameters are updated using the REINFORCE algorithm with baseline:

$$\theta_a \leftarrow \theta_a + \alpha[\frac{1}{B} \sum_{j=1}^{B} \nabla_{\theta_a} \log \pi_{\theta_a} (R^j - \hat{V}^j_{\theta_c}(S_0))] \tag{8}$$

Some studies [31] use a greedy rollout of the actor with the current set of parameters to avoid the complexities of training the critic network for Equation (7).

To learn stochastic policy π_θ using an actor network, an encoder-decoder structure originated from machine translation has shown its effectiveness due to the similar nature of the problems in both fields [32]. For instance, both machine translation and VRPs aim to construct a sequence of words or nodes, given the initial set of words or nodes. The only difference is that the input set of words in machine translation also has a sequential nature. Therefore, an *encoder* used in

modeling VRPs differs from its original setting in machine translation and serves as a graph embedding responsible for passing the graph structures efficiently to a decoder [9].

Encoders maybe static and dynamic in nature. Static encoders embed a given initial graph only once, embedding both static v_s and dynamic v_d, elements of the nodes:

$$h_v^s = F^s(v_s), \quad h_d^v = F^d(v_d) \quad \forall v \in \mathbb{V}. \tag{9}$$

where F^s, F^d are series of nonlinear functions, and h_v^s and h_v^d are the embeddings of the static and dynamic elements of node v. On the other hand, dynamic encoders embed the graph at each time t reflecting the changes due to the routing decisions on the dynamic parts of the nodes:

$$h_v^{d_t} = F^d(v_{d_t}) \quad \forall v \in \mathbb{V}. \tag{10}$$

The final embedding of a graph $\hat{h}_t$ is computed as the mean of embedding of all nodes in a graph at time t:

$$\hat{h}_t = \frac{1}{|\mathbb{V}|} \sum_{v \in \mathbb{V}} h_v^t \tag{11}$$

where h_v^t is the final embedding of node v. The graph embedding is the same for static encoders across all t. Therefore, static encoders are computed only once per problem instance, while dynamic encoders are invoked at each time step, resulting in considerable computational time. Research conducted in [33] and [34] has demonstrated the effectiveness of dynamic encoders in enhancing the results.

The static and dynamic features of nodes are problem-specific and may include the coordinates of the nodes, the customer demand at each node, and others. Regardless of the types of RL algorithms used, VRPs require learning efficient graph representations that transform discrete and complex information between nodes and edges in a graph into a continuous vector space. Various models have been proposed to learn graph embeddings. For instance, [32] uses RNNs as an encoder as a part of the Pointer Network, which [9] shows to be unnecessary due to the absence of the sequential relationship in the given initial set of nodes in a graph. Independently, [35] proposed a novel graph representation called Structure2Vec (S2V) that can encode both the graph and the partial solution at any time step. Reference [31] proposes fully attention-based encoder, introducing transformers [36] to solve VRPs, while [27], [37], and [38] uses GNNs, a deep learning model dedicated to learn graph information. In return, other studies adopt the proposed encoders, including Pointer Networks [39], multi-head attention [40], [41], [42], [43], [44], recurrent neural networks (RNNs) [45], S2V [46], [47], [48], and others. However, no studies focused on investigating which models are the most efficient in graph embedding to solve VRPs.

A *decoder* is a dynamic part of the actor-network invoked at each time step that incorporates the current state of a problem with the graph embedding to sample the next action. In particular, the decoder computes compatibility u_t at time step t that aims to extract relevant features from the inputs, which are later passed through softmax to produce the probabilities of selecting the next action:

$$u_t = F(\hat{h}_t, F(s_t)), \quad \pi_\theta(a|s) = \text{softmax}(u_t) \tag{12}$$

where F can be a series of nonlinear transformations. RNNs, which enable the passing of information about the sequence of the partial solution, have been a common choice for a decoder that is followed by additive attention [49] to compute the alignment score between the graph embedding and the current state [9], [32], [50]. A fully attention-based model is another popular design choice for a decoder, where multi-head attention is used to compute the compatibility between the context node representing the state and the graph embedding [31], [33], [41], [51], [52], [53]. Recently, [54] introduced a novel idea of a feature embedding refiner between the encoder and decoder to boost existing models.

Table II summarizes studies with end-to-end RL methods, where due to the episodic nature of VRPs REINFORCE algorithm with a baseline is used. However, the choice of the baseline varies across studies. For instance, [9], [32], [46] and others train a separate neural network called the critic to estimate the baseline, total expected reward, given the initial state of a graph. On the other hand, [31], [42], and others use a training model in a greedy setting to estimate the baseline value. Other policy gradient methods such as Deterministic Policy Gradient and Actor-critic methods have been proposed in [45] and [55]. However, the greedy rollout resulting from a trained end-to-end RL method may not be competitive against handcrafted heuristics. Therefore, it is common to use different search methods with the trained models. One of them is *active search*, which aims to adopt the trained model parameters to the specifics of an instance, thus improving the solution quality [56]. Another search method is *random sampling*, indicating the number of total solutions to be sampled, among them, we select the best solution for a reward.

III. Hybrid Methods

Hybrid methods are a combination of both learning and non-learning methods to address complex VRPs. Based on the structure of the hybrid method solutions, they can be classified into two groups. The first group involves using learning methods as a supporting tool to address the internal subproblems of non-learning methods. This approach can enhance the efficiency of non-learning methods by leveraging the learning methods to tackle specific subproblems. In the second group, learning methods are the primary tool to produce a solution, which is then improved with construction heuristics. Additionally, optimization layers can be added to learn the costs [63], [64] for training machine learning models. By combining learning methods with construction heuristics, hybrid methods can produce efficient solutions for VRPs. Overall, hybrid methods are a promising approach for addressing the challenges of VRPs by taking advantage of the strengths of both learning and non-learning methods.

A. Learning Methods for Facilitating Non-Learning Methods

Combining Q-learning with meta-heuristics dates back to [65], which proposed to use ant-colony in a distributed setting

TABLE II

THE SUMMARY OF STUDIES WITH END-TO-END RL METHODS

CSP-COVERING SALESMAN PROBLEM, DVRP-DYNAMIC VRP, PCTSP-PRIZE COLLECTING TSP, SDVRP-SPLIT DELIVERY VRP, SPCTSP-STOCHASTIC PRIZE COLLECTING TSP, OP-ORIENTEERING PROBLEM, MCVRP-MULTIPLE CVRP, MVRPSTW- MULTIPLE VRP WITH SOFT TIME WINDOWS, MAMP-MULTI-AGENT MAPPING PROBLEM, MOTSP-MULTI-OBJECTIVE TSP, MCVRPTW-MULTIPLE CVRP WITH TIME WINDOWS, EVRPTW-ELECTRICAL VEHICLE ROUTING PROBLEM WITH TIME WINDOWS, PDP-PICKUP AND DELIVERY PROBLEM, HCVRP-HETEROGENEOUS CVRP, DU-VRP-DYNAMIC AND UNCERTAIN VRP

Study	Problems	Training Algorithm	Graph Embedding	Decoder
Bello et al. [32]	TSP	REINFORCE with baseline	Pointer Network	Pointer Network
James et al. [46]	DVRP	REINFORCE with baseline	S2V	Pointer Network
Dai et al. [28]	TSP	DQN	S2V	-
Nazari et al. [9]	TSP, CVRP	REINFORCE with baseline	Elementwise projections	RNN with Attention
Xin et al. [34]	TSP, CVRP	REINFORCE with the greedy rollout	Pointer Network, Multi-head attention (Dynamic)	Pointer Network, Multi-head attention
Vera and Abad [50]	mCVRP	Actor-critic (A2C)	RNN	RNN with Attention
Zhang et al. [40]	mVRPSTW	REINFORCE with baseline	Multi-head attention	Multi-head attention
Kool et al. [31]	TSP, CVRP, OP, PCTSP, SDVRP, SPCTSP	REINFORCE with baseline	Multi-head attention	Multi-head attention
Peng et al. [33]	CVRP	REINFORCE with baseline	Graph Attention Network (GAT)	Multi-head attention
Sykora et al. [57]	MAMP	REINFORCE	Linear projection	RNN with attention
Xin et al. [41]	TSP, CVRP, OP, PCTSP, SDVRP, SPCTSP	REINFORCE with the greedy rollout	Multi-head attention	Multi-head attention
Xu et al. [58]	TSP, CVRP, PCTSP, SDVRP	REINFORCE with baseline	Multi-head attention with gate aggregation	Self-attention with a attentive aggregation module
Kim et al. [42]	TSP, CVRP, PCTSP	REINFORCE with greedy rollout	Multi-head attention	Multi-head attention
Kwon et al. [43]	TSP, CVRP	REINFORCE with baseline	Multi-head attention	Multi-head attention
Emami and Ranka [45]	TSP	Deterministic Policy Gradient (DPG)	RNN	Sinkhorn layer
Joshi et al. [26]	TSP	REINFORCE with greedy rollout	Multi-head attention	Multi-head attention
Falkner and Schmidt-Thieme [51]	mCVRPTW	REINFORCE with greedy rollout	Self-attention, Linear projection	Multi-head attention
Joshi et al. [27]	TSP	REINFORCE with baseline	GNN	Multi-head attention
Li et al. [39]	CSP	REINFORCE with the greedy rollout	Pointer Network	RNN with attention
Drori et al. [59]	TSP, VRP	REINFORCE with baseline	GAT	Multi-head attention
Lin et al. [47]	EVRPTW	REINFORCE with greedy rollout	S2V	RNN with Attention
Li et al. [55]	MOTSP	Actor-critic (A2C)	1D-Conv	Gated recurrent unit (GRU) with Attention
Li et al. [60]	PDP	REINFORCE with greedy rollout	Multi-head attention	Multi-head attention
Li et al. [61]	HCVRP	REINFORCE with greedy rollout	Multi-head attention	Feed-Forward networks, Multi-head attention
Duan et al. [62]	CVRP	REINFORCE with greedy rollout, SUPERVISE with policy-sampling	Graph Convolutional Networks	GRU with context-based attention, Multi-layer Perceptron
Pan and Liu [48]	DU-VRP	REINFORCE with baseline	S2V	RNN with Attention

TABLE II

(Continued.) THE SUMMARY OF STUDIES WITH END-TO-END RL METHODS

CSP-COVERING SALESMAN PROBLEM, DVRP-DYNAMIC VRP, PCTSP-PRIZE COLLECTING TSP, SDVRP-SPLIT DELIVERY VRP, SPCTSP-STOCHASTIC PRIZE COLLECTING TSP, OP-ORIENTEERING PROBLEM, MCVRP-MULTIPLE CVRP, MVRPSTW-MULTIPLE VRP WITH SOFT TIME WINDOWS, MAMP-MULTI-AGENT MAPPING PROBLEM, MOTSP-MULTI-OBJECTIVE TSP, MCVRPTW-MULTIPLE CVRP WITH TIME WINDOWS, EVRPTW-ELECTRICAL VEHICLE ROUTING PROBLEM WITH TIME WINDOWS, PDP-PICKUP AND DELIVERY PROBLEM, HCVRP-HETEROGENEOUS CVRP, DU-VRP-DYNAMIC AND UNCERTAIN VRP

Lei et al. [52]	TSP, CVRP	REINFORCE with baseline	E-GAT	Pointer Network, Transformer Network
Gunarathna et al. [53]	VRP, TSP and their dynamic versions	REINFORCE	Temporal Encoder motivated by d STAtt Block	Multihead-attention, pointer network
Jiang et al. [37]	TSP, VRP	REINFORCE	distributionally robust optimization and CNN	Transformers
Senuma et al. [38]	TSP, VRP	REINFORCE with baseline	Graph Convolutional Networks	Pointer Networks
Goh et al. [44]	TSP	REINFORCE	Transformer	Sinkhorn layer

to explore solutions for TSP. In particular, each ant represents an agent to construct solutions and fill a Q-table associated with moving from one city to another. At the same time, the reward function reinforces selecting tours with short lengths. Later, [66] further developed the idea of enhancing cooperation between ants/agents through the introduction of updated pheromone values assigned to each edge in the graph. The study has shown that combining the ant-colony system with local search achieves competitive results in solving TSP.

Reference [67] combines the genetic algorithm with RL to solve TSP, where the Q-learning algorithm is modified along with the mutation operation to produce efficient tours, which later can be enforced with local search. Similarly, [68] combines the genetic algorithm with multi-agent RL, where MARL is used to generate an initial solution which later will be improved by a genetic algorithm. Reference [69] integrated RL into the NSGA-II optimization algorithm by using the Q-learning algorithm for local search. This approach enables the algorithm to generate quality solutions by improving its search process. The authors showed that the use of Q-learning is able to achieve superior performance compared to the Q-learning-based multiobjective evolutionary algorithms with the randomly selected neighborhood. Reference [70] proposed an alternative approach to enhance evolutionary algorithms by utilizing a radial basis function network (RBFN) for population evaluation. The study highlights that a well-trained RBFN can provide a remarkable boost to the overall performance and can expedite the convergence of the proposed model.

Reference [71] presents a model to use RL for neighborhood search (NS) to solve CVRP and split delivery VRP (SDVRP). In particular, a feasible solution is destroyed by a destroy operator in several locations. Next, a repair operator is trained using RL to connect the partial solutions. In particular, an encoder-decoder model is proposed to learn the embeddings of the end nodes of the partial solutions and produce the probabilities of connecting nodes to construct a complete solution. The results outperform [9], [31] for VRP and SDVRP for different node sizes and show competitive results with LKH3, a unified hybrid genetic search. Also, [72] proposes using [9] model combined with Large Neighborhood Search (LNS) to solve VRPTW in ride-hailing services. They train the neural networks using SL for the insertion operation inside LNS. Reference [73] further enhances the idea of [74] using multi-agent systems to learn which meta-heuristic to apply to solve VRPTW and develops adaptive local search through Q-learning. Q-learning aims to determine the sequence of neighborhoods that need to be explored first to maximize the objective function by picking the actions as different operations available from LNS.

Along with approximated DP methods used to solve VRPs [75], [76], the combination of RL with DP is an emerging research direction. For instance, [77] proposes to combine RL with constraint programming (CP) using a DP approach to solve TSP, VRP, and TSPTW. Deep Q-learning and Proximal Policy Optimization (PPO) are integrated into CP search algorithms. Reference [78] aims to mitigate the curse of dimensionality present in DP by restricting its search space to policies produced by learning methods. Identical to [23], the study generates the heatmap for the edges of a given graph that indicates the probabilities of entering those edges into the solution. Afterwards, this heatmap is used for branching in forwarding DP. References [79] and [80] use neural networks to approximate the value functions for DP, which expedites the solution time. References [81] utilizes a policy iteration algorithm to solve CVRP. The authors propose a simple model consisting of fully connected hidden layers and a ReLU activation function to estimate the value of each state defined as the set of unvisited nodes. Additionally, or policy evaluation, they used MIP to select actions, combining the learned value functions. The resulting MIP is solved using the branch-and-cut approach of [82].

Furthermore, clustering algorithms have also been applied to routing problems. These algorithms aim to split the problem into smaller sub-problems and solve them using non-learning methods. In particular, several recent studies [83], [84], [85], [86], [87] have utilized clustering algorithms to group customers. The resulting clusters are then provided to

non-learning methods to generate solutions considering certain constraints, such as time windows.

Another set of studies uses learning methods to pick heuristics to solve routing problems. For instance, [88], [89] aim to learn improved heuristics. They embed the nodes and positions in a given solution separately, wherein in the middle of the encoding, they are shared with each other, but at the end, the encoder produces different node features and position feature embedding that is passed to the decoder. The decoder produces the matrix of probabilities to select the pair of nodes to use for local operation. Reference [90] proposes the NeuRewriter model, that given an initial solution, picks the region to be modified and applies some learned rewriting rules to improve the solution. Both region-picker and rule-picker policies are trained using RL to solve TSP and CVRP. Similarly, [91] proposes to learn a general heuristic to solve VRP, VRPTW that given an initial solution destroys some part of the solution and learns to reconstruct an improved version of it. The study modifies GNNs to embed both the nodes and edges, which passed through GRU to define the order of the nodes of the newly repaired part for the solution. Also, [92] proposes to delegate subproblems of VRPs to black-box solvers and trains a subproblem generator model based on transformers and SL. In particular, they restrict the possible number of subproblems to solve given an initial solution by looking at only subsets of the visited nodes and restricting the number of routes to be considered by utilizing spatial locality.

B. Improving Learning Methods With Heuristics

Along with sampling methods such as beam search, active search, and random sampling, trained learning methods can be combined with traditional construction heuristics. In particular, a greedy solution produced by a learning method is used as an initial solution. For instance, [93] solves TSP with the MHA-based encoder and the Pointer Network decoder along with a 2-opt local search. For solving VRP and VRPTW on a large scale, [94] used [9]' model with an adaptive critic later combined with local search algorithms such as OR-tools and LNS. Also, [95] introduces Graph Pointer Nets (GPNs) to solve TSP, where a graph encoder made of GNNs is combined with Pointer Networks. In light of this, the paper proposes a Hierarchical RL to solve TSPTW, where each layer of the neural network learns to solve TSP with some constraint. By combining GPNs with local search, they obtained comparable results with the construction heuristics. They also search real and random data instances of TSPTW and demonstrate comparable results with the construction heuristics. Another study by [96] solves TSP with Time Windows and Rejections, where the objective is to minimize the rejection and distance costs. The paper proposes to use the AM model to solve the regular TSP and deploy a heuristic that will check the feasibility of the produced solution according to Time Windows and determine if an order has been rejected. This will help compute the rewards function correctly according to TSPTWR and update the parameters of NNs using the baseline rollout. Recently, [97] updated their model [88] combined with NS to solve the Pickup and Delivery Traveling Salesman Problem (PDTSP) more efficiently. They use Neural Neighborhood Search to tackle the precedence constraint quickly by allowing a pair of pickup-delivery nodes to be simultaneously operated in the neighborhood search through two customized removal and reinsertion decoders.

Learning methods can also be combined with prescriptive methods to solve complex routing problems. For instance, [98] discusses the problem of managing a large set of available homogeneous vehicles for online ride-sharing platforms. In this work, the authors propose Contextual DQN and Contextual Actor-Critic networks to learn a state-value function shared by all agents representing each vehicle. The learned state-value functions are used for efficient allocation solved by linear programming.

Another set of studies combines learning methods with Monte-Carlo Tree Search (MCTS). For instance, [99] proposes the idea of training TSP in a supervised manner in small instances using LKH as the optimal solution and then subsequently utilizing the trained model to solve large instances. First, the authors train a graph convolutional residual network with an attention mechanism on a graph with m nodes. Furthermore, they take a larger graph and split it into several graphs with size m, where each node can be present in several graphs. Consequently, they use the trained model for each subgraph to generate the heatmap that shows the probabilities of connecting nodes in the graph. To combine the heatmap and produce a single solution, they sum the probabilities for each arc computed across all the subgraphs, and this summed heatmap is used for MCTS to further improve the solution, similar to [100]. The study solves TSPs with up to 10,000 nodes in a reasonable time. Table III summarizes studies with hybrid approaches, encompassing a wide variety of learning methods, including supervising learning, Q-learning, clustering, and policy gradient methods in combination with MIPs, heuristics, and metaheuristics.

IV. Single vs. Multiple VRP Formulations

The majority of VRPs in practice involve routing either multiple heterogeneous or homogeneous vehicles [115]. However, the straightforward application of learning methods to such VRPs is challenging due to the presence of several vehicles in a graph that all can influence the shared environment. This section presents the MDP formulations for routing a single vehicle using CVRP as an example. In addition to the formulation, we present the comparison results for TSP, CVRP, and VRPTW. Finally, we discuss several approaches present in the literature to formulate multiple VRPs.

A. Single Vehicle Routing Problems

A single vehicle routing problem is defined in a graph $\mathbb{G}(\mathbb{V}, \mathbb{E})$ consisting of a set of nodes $\mathbb{V} = \{v_0, \ldots, v_n\}$ and a set of edges $\mathbb{E} = \{(v_i, v_j) : i < j, v_i, v_j \in \mathbb{V}\}$, where each node represents depot v_0 or customer's locations $\mathbb{V} \setminus v_0$. The objective is to find a sequence of nodes to be visited by a vehicle to minimize the total costs, defined as the total distance traveled. Capacity constraints can be added by setting the maximum load $l_{\max}$ to be carried by a vehicle, with d_v

TABLE III

THE SUMMARY OF STUDIES WITH HYBRID METHODS DISCUSSED IN SECTION IV
CVRP-CAPACITATED VRP, CVRPTW-CVRP WITH TIME WINDOWS, MARL-MULTI-AGENT RL, SDVRP-SPLIT DELIVERY VRP, TSPTW-TSP WITH TIME WINDOWS, TSPTWR-TSP WITH TIME WINDOWS AND REJECTION, VRPTW-VRP WITH TIME WINDOWS, SDDPVD-SAME-DAY DELIVERY PROBLEM WITH VEHICLES AND DRONES, MVRPTW- MULTIPLE VRP WITH TIME WINDOWS, DVRP-DYNAMIC VRP, ATSP-ASYMMETRIC TSP, CMVRPTWMDP-COLLABORATIVE MULTICENTER VRP WITH TIME WINDOWS AND MIXED DELIVERIES AND PICKUPS, TDGVRPTW-TIME-DEPENDENT GREEN VRP WITH TIME WINDOWS, MO-VRPSD-MULTI-OBJECTIVE VRP WITH STOCHASTIC DEMAND, FDTSP-FLEXIBLE DRONES TSP

Name	Problems	Type	Learning Method	Non-learning method
Dorigo and Gambardella [66]	TSP	First	Q-Learning	Ant-colony
Liu and Zeng [67]	TSP	First	Q-learning	Genetic Algorithm
Alipour et al. [68]	TSP	First	MARL	Genetic Algorithm, 2-opt
Hottung and Tierney [71]	CVRP, SDVRP	First	Policy-based RL	NS
Syed et al. [72]	VRPTW in ride-hailing services	First	SL	Large Neighborhood Search
Fernandes et al. [74]	VRPTW	First	Q-learning	Adaptive Local Search
Cappart et al. [77]	TSPTW	First	DQN, PPO	CP, DP
Kool et al. [78]	TSP, CVRP, TSPTW	First	SL	DP
Xu et al. [79]	TSP	First	Unsupervised Learning	DP
Delarue et al. [81]	CVRP	First	Value-based RL	Mixed Integer Programming (MIP)
Ma et al. [88]	CVRP	First	The actor-critic variant of PPO	2-opt, swap and insert
Chen and Tian [90]	CVRP	First	Actor-Critic	Halide rewriter [101]
Li et al. [92]	Large-scale CVRP	First	SL	MIP solver
Deudon et al. [93]	TSP	Second	MHA-based encoder, Transformers Decoder	2-opt local search
Zhao et al. [94]	CVRP and VRPTW in large scale	Second	Policy Gradient	OR-tools and LNS
Ma et al. [95]	TSPTW	Second	Hierarchical Policy based RL	Local search
Zhang et al. [96]	TSPTWR	Second	Policy based RL	Tabu search
Lin et al. [98]	Large-scale fleet management problem	Second	Contextual DQN and Contextual Actor-Critic	Linear programming
Chen et al. [102]	SDDPVD	First	Deep Q-learning	A policy function approximation
Tyasnurita et al. [103]	Open VRP	First	Hyper-heuristic classification	Modified Choice Function All Moves
Xing and Tu [100]	TSP	Second	SL to learn heat maps	Monte Carlo Tree Search
Fu et al. [99]	TSP	Second	SL to learn heat maps	Monte Carlo Tree Search
Hottung et al. [104]	TSP, CVRP	Second	SL to learn latent space	Unconstrained continuous optimization
Gutierrez-Rodríguez et al. [105]	mVRPTW	Second	SL	HMOEA-06 [106], MOGA-06 [107], MMOEAD-15 [108], HMPSO-16 [109]
da Costa et al. [35]	TSP	Second	DRL, Policy Gradient	2-opt local search
Fu et al. [110]	TSP	First	MCTS with RL	2-opt local search
Wu et al. [89]	TSP, CVRP	First	Policy gradient	Pairwise local operators
Xin et al. [111]	TSP, CVRP, CVRPTW	First	SL	LKH Algorithm
Yang et al. [80]	TSP	First	Deep Q-Learning	DP
Gao et al. [91]	CVRP, CVRPTW	Second	Actor-Critic	Very Large-scale Neighborhood Search
Joe and Lau [112]	DVRP	First	TD learning	Simulated Annealing

TABLE III
(Continued.) THE SUMMARY OF STUDIES WITH HYBRID METHODS DISCUSSED IN SECTION IV
CVRP-CAPACITATED VRP, CVRPTW-CVRP WITH TIME WINDOWS, MARL-MULTI-AGENT RL, SDVRP-SPLIT DELIVERY VRP, TSPTW-TSP WITH TIME WINDOWS, TSPTWR-TSP WITH TIME WINDOWS AND REJECTION, VRPTW-VRP WITH TIME WINDOWS, SDDPVD-SAME-DAY DELIVERY PROBLEM WITH VEHICLES AND DRONES, MVRPTW- MULTIPLE VRP WITH TIME WINDOWS, DVRP-DYNAMIC VRP, ATSP-ASYMMETRIC TSP, CMVRPTWMDP-COLLABORATIVE MULTICENTER VRP WITH TIME WINDOWS AND MIXED DELIVERIES AND PICKUPS, TDGVRPTW-TIME-DEPENDENT GREEN VRP WITH TIME WINDOWS, MO-VRPSD-MULTI-OBJECTIVE VRP WITH STOCHASTIC DEMAND, FDTSP-FLEXIBLE DRONES TSP

Gambardella and Dorigo [65]	TSP, ATSP	First	Q-learning	Ant System
Wang et al. [84]	CMVRPTWMDP	First	Improved 3D k-means clustering algorithm	genetic algorithm, particle swarm optimization
Qi et al. [69]	TDGVRPTW	First	Q-learning	updated NSGA-II
Zong et al. [113]	CVRP	First	PCA, LSTM, Generator	HGS
Niu et al. [70]	VRPSD, MO-VRPSD	First	RBFN	multi-objective evolution algorithm with ENS-SS
Lu et al. [85]	FDTSP	First	K-means clustering	Or-Tools
Ma et al. [97]	PDTSP	Second	Transformer based Encoder-Decoder	Neighborhood Search
Zheng et al. [114]	TSP	First	Q-learning	LKH

representing the customer demand at node v. The problem naturally fits a single-agent RL, where a vehicle represents the agent, and the environment is a simulated CVRP. The agent is trained to select action $A_t \in \mathbb{V}$ at each time step t, representing the node index to be visited next given the current state S_t. A good state representation should fully capture the dynamics of the problem and make future decisions based only on the current state, thus preserving the Markov property. A common choice of a state for CVRP is defined by a tuple $S_t = \{a_t, l_t, Y_t\}$, representing the current location of a vehicle, the remaining load, and the set of served customers [9], [31]. At each time step t, we can update S_{t+1} given the current state S_t and selected action a_t.

After each decision at time t, we calculate intermediate reward r_t, defined as the distance between the current location a_t and the node to be visited a_{t+1}, from which we can derive the total reward $R = \sum_{t=0}^{T} r_t$. Single vehicle routing problems, involving routing decisions of only a single vehicle, may come with modifications, such as considering time window constraints [74], [94], [111], allowing split deliveries [9], [71], introducing dynamic demand [46], or generalizing to various forms of TSP [31], [41], [95], [96].

B. Multi-Vehicle Routing

Multiple vehicle routing problems can also be modeled using the above single-agent RL formulation. For example, when routing M homogeneous vehicles to minimize the total distance traveled, the trained RL agent solution can be split into M vehicles, where each vehicle is allowed to return to the depot only once. However, in VRPs, all vehicles share a common goal of serving customers with the least total costs. Therefore, the idea of using *a centralized controller*, responsible for routing all vehicles in coordination to achieve a common goal, has been used in many studies.

In *a centralized controller* setting, the controller is the agent, taking actions for all vehicles at each time step. Formally, the central controller has action state $\mathbb{A}_{t+1} = \{A^1_{t+1}, \ldots, A^M_{t+1}\}$, where M is the number of vehicles and $A^m_{t+1} \in \mathbb{V}$ for all $m = 1, \ldots, M$. The current state $S_t = \{\mathbb{A}_t, \mathbf{l_t}, Y_t\}$ consists of a tuple indicating the last selected actions of all vehicles, the vector of remaining loads for each vehicle, and the set of customers served by any vehicle. The ensuing step involves training the central controller to efficiently construct routes for all vehicles denoted as $p = [p_0, \ldots, p_T]$, where $p_t = [a^1_t, \ldots, a^M_t]$. Subsequently, for each vehicle m, we have a route $p^m = [a^m_0, \ldots a^m_T]$. In case when the objective is to minimize the total time spent to serve all customers, the central controller aims to minimize the total reward defined as:

$$R = \min_{p}\{\max\{c^1(p^1), \ldots, c^m(p^m)\}\} \tag{13}$$

where $c^m(p^m) = \sum_{t=0}^{T-1} \Delta t^m_{a^m_t, a^m_{t+1}}$, where $\Delta t^m_{a^m_t, a^m_{t+1}}$ is the traveling time of vehicle m from node a^m_t to node a^m_{t+1}. Unlike the MARL formulation, the central controller observes the entire environment and has full knowledge about the state of each vehicle, allowing it to promote coordinated routing for both homogeneous fleets and heterogeneous vehicles. For instance, all vehicles have common action space for multiple vehicles operating in a shared graph. Without coordination, several vehicles may visit the same customer. The central controller, which assigns actions for each vehicle, prevents such inefficiencies. Studies such as [8], [47], [50], [96], [105], [116], and [117] presented in Table IV use the centralized controller to solve mCVRP and mVRPTW with a homogeneous set of vehicles. Additionally, [46] and [98] propose deep RL to route multiple vehicles with uncertain demand. Another emerging topic is to route a set of heterogeneous vehicles such as trucks and drones [90], [118], also relying on the centralized controller.

TABLE IV
THE SUMMARY OF STUDIES TO SOLVE MULTI-VEHICLE ROUTING PROBLEMS. DSCVRPTW - DYNAMIC AND STOCHASTIC CVRPTW, SCVRPTW-STOCHASTIC CVRPTW, MMHCVRP-MIN-MAX HETEROGENEOUS CVRP, MSHCVR-MIN-SUM HETEROGENEOUS CVRP

Name	Problems	Multi-agent	Central Controller	Hetero-geneous	Homo-geneous	Offline	Online
James et al. [46]	Online DVRP		✓	✓			✓
Vera and Abad [50]	mCVRP		✓	✓		✓	
Zhang et al. [40]	mVRP with soft TW		✓		✓	✓	
Sykora et al. [57]	Multi-agent Mapping Problem	✓			✓	✓	
Lin et al. [98]	online ride-sharing		✓		✓		✓
Falkner and Schmidt-Thieme [51]	mCVRPTW		✓		✓	✓	
Qin et al. [117]	VRP		✓	✓		✓	
Silva et al. [73]	mVRPTW	✓			✓	✓	
Van Knippenberg et al. [24]	mCVRP		✓		✓	✓	
Bogyrbayeva et al. [8]	mCVRP with charging		✓		✓	✓	
Bogyrbayeva et al. [118]	TSP with Drone		✓	✓		✓	
Chen et al. [102]	SDDPVD		✓	✓			✓
Lin et al. [47]	mEVRPTW		✓		✓	✓	
Gutierrez-Rodríguez et al. [105]	mVRPTW		✓		✓	✓	
Bono et al. [119]	mDSCVRPTW, mSCVRPTW		✓		✓		✓
Li et al. [61]	MMHCVRP, MSHCVRP		✓	✓			✓
Banfi et al. [120]	multi-robot heterogeneous area inspection problem		✓		✓	✓	

On the other hand, MARL aims to train fully independent agents which cooperate for a common goal. Formally, each agent m has its local observation denoted as o_t^m, which includes partial information about the environment. Each agent is trained to select action A_{t+1}^m given its observation o_t^m and its state s_t^m. To promote coordinated routing, communication messages are allowed between agents. Each agent is trained to construct its route $p^m = [a_0^m, \ldots, a_T^m]$, contributing to the common reward $R = \sum_{m=1}^{M} c^m(p^m)$, where $c^m(p^m)$ is a cost function. The general framework to apply MARL for routing multiple vehicles is presented in [73]. MARL formulations are suitable in dynamic environments when customer demand is observed over time. For instance, in the case of online routing, vehicles trained in a MARL setting can make independent decisions to serve customers. An example is [57], presents the MARL formulation to solve Multi-agent Mapping Problem to satisfy demand in known locations with unknown quantities.

V. COMPUTATIONAL STUDIES

In this section, we present the experimental results obtained from evaluating the performances of various models.

Data Generation: To compare the models presented to solve TSP and VRP in random data, we used the benchmark dataset presented in [31], where coordinates of the customer locations are generated using uniform distribution in a unit square. For CVRP, the customer demand at each node is an integer random variable sampled uniformly with values ranging from 1 to 9, and the capacity is set as 30, 40, and 50 for the graphs with 20, 50, and 100 nodes, respectively.

Baselines: We compare end-to-end learning methods [31], [41], [42], [43], hybrid methods [88], [89], [104], [121] in their different configurations to solve CVRP on different graph sizes and use LKH3 as baselines. The implementations and the pre-trained models are taken from the respective shared GitHub repositories. Even though a model-agnostic refiner proposed in [54] shows promising results in boosting [31], [43], in the experiments, we focus on the original models.

Run times: We run our experiments using a single GPU (RTX2080) and 16 cores of CPU (Intel Core i9-9900K 3.60GHz) on Ubuntu 20.04. We set the batch size equal to one in both greedy and sampling decoding strategies if the shared code of the models can support such a setting.

Results: Table V presents the performance comparisons, where the objective represents the average cost of solving 1,000 instances of CVRP. The gap is computed as follows: $GAP = \frac{Z-Z^*}{Z^*}$, where Z^* is the cost of LKH3 algorithm, and Z is the cost of a method. We report the average of the gaps and the total time spent across all test instances. Among learning-based models, the hybrid model of [88] with data augmentation performs slightly better than the rest of the presented studies, but it requires a long running time. CVRP is a challenging problem, where the gaps between learning-based methods and optimization-based algorithms such as LKH3 still exists. However, [43] can solve instances on large sizes with 1.27% gap on average in a fraction of time compared to LKH3.

Even though there have been some studies to solve VRPTW [40], [74], [94], the settings of the problem are different for each study, preventing the comparisons between them. Therefore, in Table VI, we present the comparison between [31] and [51] on the well-known Solomon dataset [122] originally reported in [51]. Table VII, we present the generalization capacity of models on well-known CVRPLib dataset, while OR-Tools [123] is used as a benchmark method to represent the optimization methods. Since all the studies have been evaluated on the same instances, we present the comparison results based on the corresponding papers. In particular, the results of [31] and [43] are based on the report from [88], the results of [121] are based on the reports from [42] and we report the results of [42], [88], and [89] from the original studies. As shown in the tables, on average, [95] performs the best, outperforming OR-Tools and other learning-based methods.

VI. THE FUTURE RESEARCH DIRECTIONS

Machine learning-based methods for VRPs can provide benefits, such as improved efficiency, real-time decision-making,

TABLE V
THE PERFORMANCE COMPARISON TO SOLVE CVRP ON RANDOM INSTANCES

Method	CVRP20			CVRP50			CVRP100		
	Cost	Gap	Time	Cost	Gap	Time	Cost	Gap	Time
LKH3	6.16	0.00%	2h18m	10.46	0.00%	9h10m	15.68	0.00%	17h27m
Kool et al. [31], AM{M: 1280}	6.28	1.93%	26s	10.71	2.42%	1m22s	16.24	3.62%	3m40s
Kool et al. [31], AM{M: 2560}	6.28	1.83%	40s	10.70	2.29%	1m56s	16.21	3.44%	5m39s
Kool et al. [31], AM{M: 7500}	6.27	1.67%	1m3s	10.68	2.09%	4m4s	16.17	3.18%	14m8s
Kwon et al. [43]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	15.87	1.27%	10s
Kwon et al. [43]×8 augment	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	15.87	1.27%	4hr23m
Ma et al. [88] (T=1k)	6.57	6.35%	31s[2]	12.82	22.46%	1m23s[2]	16.30	5.36%	4m8s[2]
Ma et al. [88] (T=5k)	6.28	1.94%	2m33s[2]	12.02	14.78%	4m[2]	15.97	3.25%	20m43s[2]
Ma et al. [88] (T=10k)	6.19	0.65%	5m12s[2]	11.73	12.01%	14m3s[2]	15.94	1.70%	49m[2]
Ma et al. [88]×6 augment	6.07	**-1.19%**	18m13s[2]	11.38	8.70%	1hr8m[2]	15.79	**0.70%**	4hr[2]
Wu et al. [89] (T=1,000)	–[1]	–[1]	–[1]	10.71	3.83%	8m4s[2]	16.27	5.16%	2hr2m[2]
Wu et al. [89] (T=3,000)	–[1]	–[1]	–[1]	10.65	3.25%	8m20s[2]	16.10	4.06%	2hr2m[2]
Wu et al. [89] (T=5,000)	–[1]	–[1]	–[1]	10.62	2.97%	8m12s[2]	16.03	3.58%	3hr36m[2]
Hottung et al. [104]	6.16	0%	4hr	10.50	**0.41%**	9hr23m	15.89	1.4%	28hr

[1] the pretrained models are not shared in the corresponding github repositories.
[2] the batch size is set to 1,000 to record running times.

TABLE VI
THE PERFORMANCE COMPARISON TO SOLVE VRPTW ON THE SOLOMON BENCHMARK PROBLEMS. K IS THE AVERAGE NUMBER OF VEHICLES, AND TINF IS THE TIME TO GENERATE A SOLUTION

	Model	VRPTW20				VRPTW50			
		Cost	K	Dist	Tinf	Cost	K	Dist	Tinf
TW1	Falkner and Schmidt-Thieme [51], OR-Tools-AU	2577.08	4.18	643.77	0.22s	4344.88	6.30	1110.68	1.76s
	Falkner and Schmidt-Thieme [51], OR-Tools-GLS	2522.85	4.11	617.77	8.00s	4213.23	6.15	1090.36	8.06s
	random (1000)	3036.39	5.68	1200.37	-	7297.53	11.80	2914.86	-
	Kool et al. [31], AM+TW(greedy)	3766.88	5.29	1264.40	0.05s	7189.43	9.42	2882.53	0.12s
	Kool et al. [31], AM+TW(sampl.)	3041.24	5.74	1202.64	0.12s	7327.09	11.92	2921.39	0.38s
	Kool et al. [31], AM+TW(t10240)	2750.06	5.27	1163.58	0.95s	6878.84	11.28	2865.30	3.08s
	Falkner and Schmidt-Thieme [51], JAMPR(greedy)	1862.40	2.25	966.74	0.10s	3055.94	5.42	1733.24	0.24s
	Falkner and Schmidt-Thieme [51], JAMPR(sampl.)	**1716.60**	2.29	965.42	0.86s	**2691.55**	4.03	1811.06	3.07s
TW2	Falkner and Schmidt-Thieme [51], OR-Tools-AU	635.06	4.23	635.06	0.37s	1123.82	6.72	1123.82	3.81s
	Falkner and Schmidt-Thieme [51], OR-Tools-GLS	**619.57**	4.14	619.21	8.30s	**1119.07**	6.67	1118.01	8.09s
	random (1000)	1646.83	6.13	1202.66	-	8368.49	8.65	2897.99	-
	Kool et al. [31], AM+TW(greedy)	7615.69	2.00	1094.39	0.05s	40245.40	2.00	2687.95	0.12s
	Kool et al. [31], AM+TW(sampl.)	1572.31	6.56	1221.09	0.11s	7712.35	9.34	2953.65	0.34s
	Kool et al. [31], AM+TW(t10240)	1387.30	6.46	1170.49	0.90s	6730.55	9.99	2954.76	2.70s
	Falkner and Schmidt-Thieme [51], JAMPR(greedy)	674.72	4.32	626.80	0.11s	1273.20	6.02	1126.74	0.25s
	Falkner and Schmidt-Thieme [51], JAMPR(sampl.)	**620.68**	4.19	602.33	0.92s	**1116.76**	5.64	1076.79	2.32s
TW3	Falkner and Schmidt-Thieme [51], OR-Tools-AU	1317.81	4.07	637.15	1.07s	2707.72	6.12	1121.57	20.63s
	Falkner and Schmidt-Thieme [51], OR-Tools-GLS	1312.71	4.11	625.80	8.00s	2753.66	6.18	1192.61	8.02s
	random (1000)	1409.35	3.34	953.48	-	4407.58	6.92	2692.78	-
	Kool et al. [31], AM+TW(greedy)	3101.21	2.00	1094.39	0.05s	26467.34	2.00	2687.95	0.12s
	Kool et al. [31], AM+TW(sampl.)	1412.16	3.42	951.92	0.12s	4161.24	7.15	2674.03	0.34s
	Kool et al. [31], AM+TW(t10240)	1318.56	3.23	899.83	0.88s	3941.47	6.77	2575.28	2.67s
	Falkner and Schmidt-Thieme [51], JAMPR(greedy)	1002.81	1.00	733.01	0.10s	3158.26	2.01	1347.72	0.23s
	Falkner and Schmidt-Thieme [51], JAMPR(sampl.)	**844.35**	1.39	660.48	0.78s	**1947.65**	2.29	1358.29	2.13s

TW1 - problems with hard time windows, TW2 - problem with soft constraint for upper bound, TW3 - problem with soft constraints for both upper and lower bounds.

customized solutions, and scalability. However, based on the presented survey, we indicate a list of future research directions that aim to further reveal the potential of machine learning methods for VRPs.

A. Generalization

The generalization of learning methods to problems of different graph sizes is one of the primary challenges. For instance, to produce competitive solutions, separate sets of models need to be trained from scratch for each graph size leading to a substantial total training time. Nevertheless, developing transfer learning from small-sized graphs to large-sized graphs has not been studied extensively except in a few studies. Reference [99] proposes to learn heatmaps for TSP on small sizes and apply them to arbitrarily large instances. Also, [27] investigates the generalization capabilities of SL and RL methods in end-to-end settings on TSP. The paper has shown that RL models have better generalization capabilities as compared to SL models. The study demonstrates that models trained on various graph sizes generalize better than the models trained on solo-sized graphs.nNotably, the recent work by [124] on SL for route optimization with robustness guarantees is a promising step forward in this area. However, there is still much work to be done to improve the efficiency of generalized solutions. At the same time, the hybrid models have demonstrated the potential to handle large-scale routing

TABLE VII

THE PERFORMANCE COMPARISON ON CVRPLIB

Instance	Depot Type	Customer Type	Wu et al. [89] (T=5k)	Ma et al. [88] (T=5k)	OR Tools	Kool et al. [31] (N=10k)	Kwon et al. [43] ×8 augment	Wu et al. [89] (T=5k, M=100)	Ma et al. [88] (T=10k)	Ma et al. [88] ×6 augment
X-n101-k25	R	R	7.70%	2.09%	6.57%	32.95%	3.64%	5.60%	1.86%	**1.47%**
X-n106-k14	E	C	4.86%	2.93%	3.72%	6.78%	**1.85%**	2.83%	2.75%	1.87%
X-n110-k13	C	R	6.39%	1.43%	7.87%	3.15%	2.05%	4.40%	0.87%	**0.13%**
X-n115-k10	C	R	13.32%	3.29%	4.50%	7.52%	3.49%	5.19%	3.26%	**1.68%**
X-n120-k6	E	RC	16.16%	3.50%	6.83%	4.54%	**2.12%**	5.56%	3.20%	2.38%
X-n125-k30	R	C	8.79%	6.51%	5.63%	35.16%	7.14%	**4.71%**	5.47%	5.44%
X-n129-k18	E	RC	11.01%	2.93%	8.37%	4.00%	**0.97%**	4.63%	2.55%	2.55%
X-n134-k13	R	C	16.06%	6.98%	21.61%	20.13%	4.22%	8.88%	5.56%	**2.63%**
X-n139-k10	C	R	14.99%	2.54%	12.02%	4.30%	2.28%	4.90%	2.16%	**2.08%**
X-n143-k7	E	R	20.20%	7.80%	11.27%	8.88%	**2.79%**	6.61%	6.47%	3.55%
X-n148-k46	R	RC	16.38%	2.69%	7.80%	79.53%	19.88%	3.60%	**2.22%**	**2.22%**
X-n153-k22	C	C	22.94%	11.06%	8.01%	78.11%	12.16%	**4.53%**	9.02%	6.53%
X-n157-k13	R	C	17.15%	4.64%	**2.57%**	16.30%	2.79%	3.60%	4.44%	3.12%
X-n162-k11	C	RC	19.16%	4.43%	6.31%	6.37%	4.77%	5.26%	3.04%	**2.62%**
X-n167-k10	E	R	18.52%	5.37%	9.34%	8.41%	4.05%	8.27%	4.28%	**3.47%**
X-n172-k51	C	RC	12.06%	6.23%	10.74%	85.37%	21.99%	4.36%	5.27%	**3.41%**
X-n176-k26	E	R	19.49%	10.29%	8.99%	20.39%	10.27%	6.16%	8.07%	**5.93%**
X-n181-k23	R	C	6.27%	3.41%	2.94%	6.45%	**2.08%**	**2.08%**	2.42%	**2.08%**
X-n186-k15	R	R	17.71%	5.99%	7.75%	6.01%	**2.15%**	7.65%	5.30%	4.94%
X-n190-k8	E	C	18.64%	7.97%	**6.53%**	46.61%	9.25%	6.78%	6.73%	6.73%
X-n195-k51	C	RC	17.04%	7.00%	13.76%	79.26%	9.23%	4.47%	4.54%	**4.36%**
X-n200-k36	R	C	9.60%	5.93%	**4.15%**	26.25%	5.01%	4.26%	5.87%	5.86%
X-n129-k18	E	RC	11.01%	2.93%	8.37%	4.00%	**0.97%**	4.63%	2.55%	2.55%
X-n134-k13	R	C	16.06%	6.98%	21.61%	20.13%	4.22%	8.88%	5.56%	**2.63%**
X-n139-k10	C	R	14.99%	2.54%	12.02%	4.30%	2.28%	4.90%	2.16%	**2.08%**
X-n143-k7	E	R	20.20%	7.80%	11.27%	8.88%	**2.79%**	6.61%	6.47%	3.55%
X-n148-k46	R	RC	16.38%	2.69%	7.80%	79.53%	19.88%	3.60%	**2.22%**	**2.22%**
X-n153-k22	C	C	22.94%	11.06%	8.01%	78.11%	12.16%	**4.53%**	9.02%	6.53%
X-n157-k13	R	C	17.15%	4.64%	**2.57%**	16.30%	2.79%	3.60%	4.44%	3.12%
X-n162-k11	C	RC	19.16%	4.43%	6.31%	6.37%	4.77%	5.26%	3.04%	**2.62%**
X-n167-k10	E	R	18.52%	5.37%	9.34%	8.41%	4.05%	8.27%	4.28%	**3.47%**
X-n172-k51	C	RC	12.06%	6.23%	10.74%	85.37%	21.99%	4.36%	5.27%	**3.41%**
X-n176-k26	E	R	19.49%	10.29%	8.99%	20.39%	10.27%	6.16%	8.07%	**5.93%**
X-n181-k23	R	C	6.27%	3.41%	2.94%	6.45%	2.08%	2.08%	2.42%	**2.08%**
X-n186-k15	R	R	17.71%	5.99%	7.75%	6.01%	**2.15%**	7.65%	5.30%	4.94%
X-n190-k8	E	C	18.64%	7.97%	**6.53%**	46.61%	9.25%	6.78%	6.73%	6.73%
X-n195-k51	C	RC	17.04%	7.00%	13.76%	79.26%	9.23%	4.47%	4.54%	**4.36%**
X-n200-k36	R	C	9.60%	5.93%	**4.15%**	26.25%	5.01%	4.26%	5.87%	5.86%
Avg. Gap for [100,150)			12.35%	3.88%	8.74%	18.81%	4.58%	5.17%	3.31%	**2.36%**
Avg. Gap for [150,200]			16.24%	6.57%	7.37%	34.50%	7.61%	5.22%	5.36%	**4.46%**
Avg. Gap for all			14.29%	5.23%	8.06%	26.66%	6.10%	5.20%	4.33%	**3.41%**

problems. For instance, [100] presents a model to solve TSP with 1,000 nodes. Reference [92] speeds up the current solvers to solve VRPs with 500 and up to 3,000 nodes. Reference [91]'s experiments indicate the competitiveness of their hybrid method with the construction heuristics for VRPTW with 400 nodes.

Presenting the performance of the proposed models on real-world datasets has become more common in recent years [8], [42], [89], [95], [118]. A particularly noteworthy advancements in this field are the recent studies conducted by [125], [126], and [127], all of which have exhibited promising outcomes. Despite these advancements, a significant amount of work is still required to enhance the efficiency of generalized solutions when applied to real-world datasets. Recently there have been some attempts [128] to establish well-accepted benchmark libraries and guidelines to compare the computational results of machine learning models with the operations research methods. However, the fair comparison of the learning-based models among themselves still remains a challenge.

B. Improving Models and Solution Methods

Complex models equipped with advanced deep neural techniques and numerous parameters to train on expensive hardware have been proposed. However, the solutions produced by learning methods in the best cases are comparable to existing non-learning methods as reported in Tables V and VI. Learning methods require the development of simple yet powerful models that can compete with traditional methods. Currently, there are strong assumptions about the inputs to the models. For instance, both end-to-end and hybrid models work with complete graphs with known coordinates of the nodes. However, practically important VRPs focus on the characteristics of the road networks such as costs of edges [129] without complete information about a graph. This indicates a strong need to develop models that can generalize to edge features only. In addition to inputs, the central question of why the presented models work remains largely unanswered. The studies present the superiority of their proposed models based on the computational studies conducted mostly on random datasets. However, the discussions over their choice of deep learning models that fit the properties of VRPs are limited. Another important avenue to explore involves addressing the challenges associated with solving VRPs that come with complex constraints. Currently, many existing end-to-end learning methods rely on a technique known as 'masking' to represent feasibility constraints. However, this approach falls short of ensuring full satisfaction of constraints. Therefore, it is crucial to look into alternative approaches that can adequately handle VRPs with complex constraints.

C. Incorporating Uncertainty and Online Routing

The power of learning models lies in their ability to generalize over data distribution. This advantage can be well exploited to incorporate dynamic elements in routing problems, which is not easy to do with the traditional methods [75]. For instance, online routing has become more critical with the increased demand for on-demand services such as parcel delivery, ride-sharing. The development of green logistic systems, including electrical, autonomous vehicles, and drones with uncertain charging times, can also be incorporated into the development of learning-based routing problems. The majority of the surveyed studies focus on routing either a single vehicle or are designed to solve the problems with additive objectives such as distance. The fundamental problem of routing multiple vehicles that share a common action space requires developing novel or proper adjustments to existing algorithms, including the development of clustering methods.

VII. Conclusion

This paper presents a survey of machine learning-based studies that can generalize to the inputs coming from the same distribution and produce quality solutions instantaneously. This potential improvement can lead to enhanced efficiency of solutions for static and dynamic VRPs in applications involving both offline and online decision-making. We present a broad taxonomy that includes both end-to-end learning and hybrid models, which use machine learning methods either to improve existing solutions or deploy them to produce initial solutions which are later improved by non-learning methods. We discuss in detail SL and RL methods for VRPs. We also present the hybrid methods for selecting heuristics, performing neighborhood searches, and estimating value and action-value functions to solve offline and online VRPs with single and multiple vehicles. In our computational studies, we compare the performances of the most recent models on random and well-known datasets.

Although there has been great progress in the machine learning literature to solve VRPs, there are many unanswered research questions, including the challenges of routing multiple vehicles in a dynamic environment and generalization of the trained models across different graph sizes. Most importantly, the development of novel models is needed that focus on the specifics of VRP solutions such as symmetry. Additionally, future research directions may include the integration of real-time data, the development of multi-objective optimization models, and the exploration of hybrid approaches combining machine learning with other optimization techniques. These advancements will be critical in enabling machine learning models to provide optimal routing solutions that consider real-world factors and can be easily implemented by logistics companies.

Appendix
Background

This section discusses two learning paradigms applied to solve VRPs, namely SL and RL. Unlike the former, RL does not rely on given labels but instead focuses on learning through interactions in order to maximize a long-term reward. A goal-oriented SL *agent* collects experiences by interacting with *environment*, where the interaction dynamics describe a task at hand. The agent aims to learn decisions that will lead to the most significant total reward through trial and error. We introduce Markov Decision Process and SL algorithms that solve sequential decision-making problems.

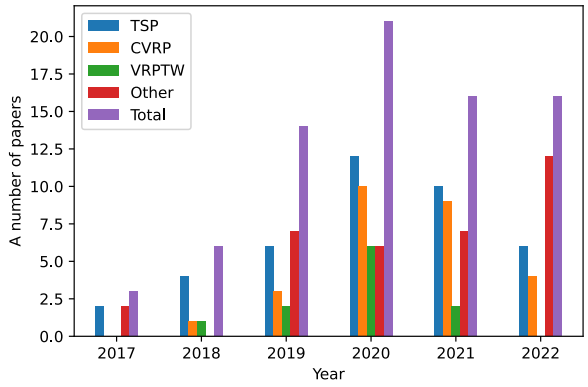

Fig. 1. A number of studies focused on solving VRPs using learning methods.

A. SL

In SL, we are given a vector of inputs, $X^T = (X_1, X_2, \ldots, X_p)$ consisting of p number of features and a vector of labels, Y. Subsequently, we can assemble a training set $D = (x_i, y_i), i = 1, \ldots, N$ and $x_i \in \mathbb{R}^p$. A SL algorithm takes input x_i and produces the predicted values of output denoted as $\hat{f}(x_i)$. The algorithm is trained by adjusting its predictions to reduce the difference between the actual output values, y_i, and its predicted values, $\hat{f}(x_i)$. Depending on the problem, we can define different loss functions to turn training parameters θ efficiently. After training, the learned function $\hat{f}(x)$ is used to predict the values of output with new values of x.

B. Markov Decision Process (MDP)

Formally, SL can be described in detail with MDP. Finite MDP consists of a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, P \rangle$ representing a finite set of states, actions, rewards, and a transition probability function, respectively. At time t, given the current state of environment S_t, an agent selects action A_t that yields a reward in the next step denoted by R_{t+1} and a new state S_{t+1}. The state-transition probability function given in 14 defines the dynamics of a problem, where the next state of the environment is only determined by the current state and action showcasing the Markov property:

$$p(s'|s,a) = P(S_{t+1} = s'|S_t = s, A_t = a). \tag{14}$$

Using the above state transition dynamics, we can determine the expected reward for the taken action a at state s with the consecutive state s':

$$\begin{aligned} r(s,a,s') &= \mathbb{E}[R_{t+1}|S_t = s, A_t = a, S_{t+1} = s'] \\ &= \frac{\sum_{r\in R} r p(s', r|a, s)}{p(s'|s,a)} \end{aligned} \tag{15}$$

In MDPs, a policy denoted as $\pi(a|s)$ maps from states to probabilities of selected actions as shown below:

$$\pi(a|s) = p(A_{t+1} = a|S_t = s). \tag{16}$$

The value function, $v(s)$, relates states to long-term rewards. A value function under policy π is formally denoted as $v_\pi(s)$, which represents the expected total reward starting from state s and following policy π over a planning horizon of T and a discount factor of γ:

$$v_\pi(s) = \mathbb{E}[\sum_{k=0}^{T-1} \gamma^k R_{t+k+1}|S_t = s] \tag{17}$$

Similarly, action-value function $q_\pi(s,a)$ defines the expected reward when taking action a at state s under policy π:

$$q_\pi(s,a) = \mathbb{E}[\sum_{k=0}^{T-1} \gamma^k R_{t+k+1}|S_t = s, A_t = a.] \tag{18}$$

We are often interested in finding a policy that leads to the most significant expected total rewards. In other words, we solve for an optimal policy π^* that yields the optimal state-value function $v^*(s)$:

$$\begin{aligned} v^*(s) &= \max_\pi v_\pi(s) \\ &= \max_a \mathbb{E}[R_{t+1} + \gamma v^*(S_{t+1})|S_t = s, A_t = a] \end{aligned} \tag{19}$$

or the optimal action-value function $q^*(s,a)$:

$$\begin{aligned} q^*(s,a) &= \max_\pi q_\pi(s,a) \\ &= \max_{a'} \mathbb{E}[R_{t+1} + \gamma q^*(S_{t+1}, a')|S_t = s, A_t = a] \end{aligned} \tag{20}$$

Equations (19) - (20) are known as the Bellman optimality equations that are recursive.

C. SL Algorithms

We can broadly define *SL* as any method that solves the above-defined MDPs. When Equations (15) and (16) are known, one can directly compute the future reward and apply planning methods. The methods that require complete knowledge about the dynamics of the problems are called *model-based*, and Dynamic Programming (DP) is one of such methods. Often Equations (15) and (16) are unknown, and one needs to learn the dynamics of the problem by learning from experience. The methods that focus directly on finding optimal state-value functions rather than requiring the complete knowledge of transition probabilities are called *model-free*. Figure 3 provides an overview of SL. Further, model-free methods can be categorized into value-based, actor-critic, and policy-based, depending on which function they aim to optimize.

Typically, in value-based methods, the goal is to find the action-value function, q^*. One of the widely used algorithms for such purpose is *Q-learning* that recursively updates the values of action and state pairs independently from a policy with α as a constant step size:

$$\begin{aligned} &Q(S_t, A_t) \\ &\leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \end{aligned} \tag{21}$$

In the simplest form shown above, Q-learning takes as a target one step Temporal-Difference (TD), $R_{t+1} + \gamma \max_a Q(S_{t+1}, a)$. The algorithm uses policy π to select actions and states whose values will be updated; however, the updates of Q-values are done through maximization over actions. To balance exploration and exploitation, Q-learning

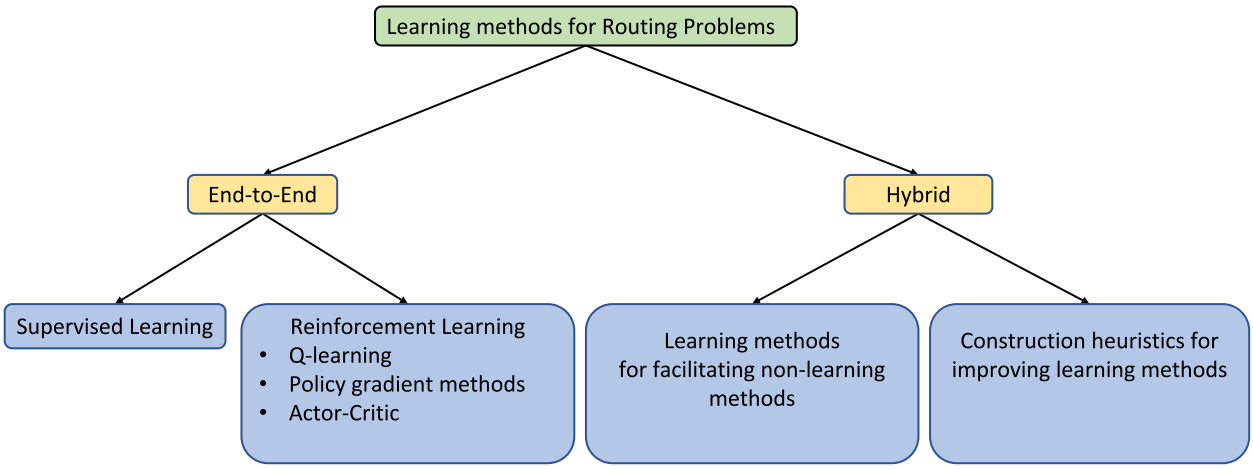

Fig. 2. The taxonomy of learning methods for VRPs.

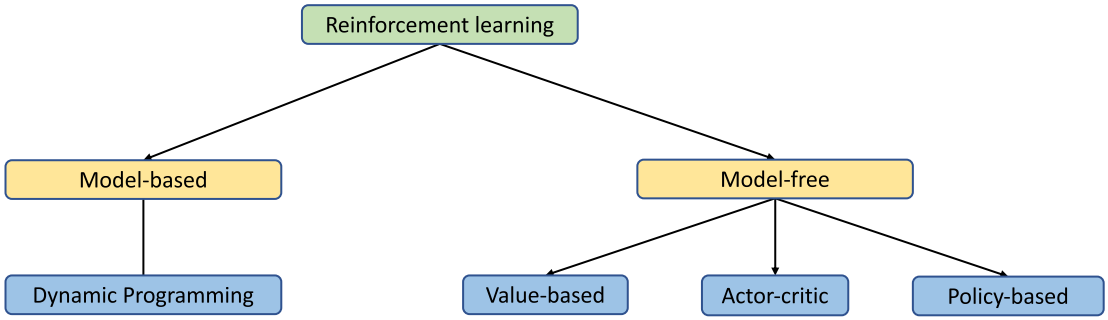

Fig. 3. An overview of SL methods.

TABLE VIII
THE PERFORMANCE COMPARISON TO SOLVE TSP ON RANDOM INSTANCES USED IN [31]. WE USED THE PRE-TRAINED MODELS FROM THE RESPECTIVE GITHUB REPOSITORIES

Method	TSP20			TSP50			TSP100		
	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
Concorde	3.84	0.00%	11s	5.69	0.00%	44s	7.75	0.00%	3m24s
LKH	3.84	0.00%	31s	5.69	0.00%	7m8s	7.75	0.00%	31m3s
Kool et al. [31], AM(N=1,280)	3.85	0.05%	18s	5.71	0.45%	1m6s	7.93	2.26%	3m11s
Kool et al. [31], AM(N=5,000)	3.85	0.05%	33s	5.71	0.38%	2m40s	7.91	2.08%	9m13s
Kim et al. [42], AM+LCP {640,10}	3.85	**0.02%**	2m10s	5.69	0.13%	3m47s	7.83	1.01%	7m13s
Kim et al. [42], AM+LCP {1280,10}	–[1]	–[1]	–[1]	5.69	0.09%	6m16s	7.82	0.88%	12m50s
Kim et al. [42], AM+LCP*{1280,45}	–[1]	–[1]	–[1]	5.70	0.16%	28m48s	7.87	2.43%	29m3s
Kwon et al. [43], POMO	3.84	0.00%	23s	5.69	0.10%	45s	7.78	0.38%	1m30s
Kwon et al. [43], POMO×8 augment	3.84	**0.00%**	19s	5.69	0.02%	46.3s	7.76	0.13%	1m54s
Xin et al. [41], MDAM-BS	3.85	**0.01%**	1hr 4m	5.70	0.20%	2hr33m	7.80	0.64%	4hr59m
Ma et al. [88], DACT (T=1k)	3.84	0.02%	16.7s[2]	5.70	0.16%	43s[2]	7.91	2.04%	2m7s[2]
Ma et al. [88], DACT(T=5k)	3.84	**0.00%**	1m22s[2]	5.69	0.02%	3m29s[2]	7.81	0.69%	10m37s[2]
Ma et al. [88], DACT(T=10k)	3.84	**0.00%**	2m48s[2]	5.69	0.01%	7m4s[2]	7.79	0.44%	21m22s[2]
Ma et al. [88], DACT×4 augment	3.84	**0.00%**	8m25s[2]	5.69	**0.00%**	25m[2]	7.76	**0.09%**	1hr25m[2]
Wu et al. [89], (T=1,000)	–[1]	–[1]	–[1]	5.73	0.81%	1m7s[2]	8	3.31%	1m34s[2]
Wu et al. [89], (T=3,000)	–[1]	–[1]	–[1]	5.70	0.29%	3m23s[2]	7.90	2.00%	4m37s[2]
Wu et al. [89], (T=5,000)	–[1]	–[1]	–[1]	5.70	0.18%	5m29s [2]	7.87	1.55%	6m
Hottung et al. [104]	3.84	**0.00%**	2hr29m	5.69	0.03%	5hr37m	7.78	0.34%	17h13m
da Costa et al. [121]	3.84	**0.00%**	3hr4m	5.69	0.12%	3hr31m	7.81	0.76%	4hr22m

[1] the pre-trained models are not shared in the corresponding GitHub repositories.
[2] the batch size is set to 1,000 to record running times.

deploys ϵ-greedy policy that selects random actions with probability ϵ and follows the greedy policy concerning the current estimates of Q with probability $1-\epsilon$.

Methods that directly focus on finding optimal policies are called policy-based methods. For instance, in policy-gradient methods, we directly aim to learn a parametrized policy π_θ

TABLE IX
THE PERFORMANCE COMPARISON ON TSPLIB

Instance	Wu et al. [89] (T=3k)	Ma et al. [88] DACT (T=3k)	OR Tools	Kool et al. [31] (N=10k)	Kwon et al. [43] POMO ×8 augment	Wu et al. [89] (T=3k, M=1k)	Ma et al. [88] (T=10k)	Ma et al. [88] ×4 augment	da Costa et al. [121]	Kim et al. [42] AM + LCP
eil51	2.82%	1.64%	2.35%	2.11%	0.00%	1.17%	**0.00%**	**0.00%**	0.23%	0.73%
berlin52	6.34%	**0.03%**	5.34%	1.67%	**0.03%**	2.57%	**0.03%**	**0.03%**	5.73%	0.10%
st70	4.59%	0.44%	1.19%	2.22%	**0.30%**	0.89%	**0.30%**	**0.30%**	0.74%	0.74%
eil76	6.88%	2.42%	4.28%	3.35%	**1.49%**	4.65%	2.04%	1.67%	2.60%	1.64%
pr76	1.40%	1.02%	2.72%	2.84%	19.97%	1.37%	**0.03%**	**0.03%**	2.60%	0.44%
rat99	17.18%	4.05%	1.73%	9.50%	7.51%	8.51%	1.16%	**0.74%**	14.62%	6.67%
KroA100	18.39%	0.86%	0.78%	79.49%	4.45%	2.08%	0.63%	**0.45%**	11.60%	2.95%
KroB100	19.97%	0.27%	3.91%	9.30%	5.83%	5.78%	**0.25%**	**0.25%**	7.45%	1.51%
KroC100	22.14%	1.06%	4.02%	8.04%	6.55%	3.17%	**0.84%**	**0.84%**	9.27%	2.84%
KroD100	16.33%	3.54%	1.61%	10.02%	8.74%	5.00%	3.54%	**0.12%**	9.58%	1.97%
KroE100	21.91%	2.17%	2.40%	3.10%	5.97%	3.29%	1.95%	**0.32%**	5.37%	1.90%
rd100	0.06%	0.08%	3.53%	1.93%	0.00%	0.06%	0.06%	**0.00%**	0.43%	0.13%
eil101	4.61%	3.66%	5.56%	3.97%	**2.07%**	4.61%	3.66%	2.86%	0.95%	2.59%
lin105	26.53%	3.41%	3.09%	32.13%	12.00%	2.48%	3.35%	**0.69%**	12.36%	3.86%
pr107	19.76%	5.86%	**1.74%**	43.26%	5.66%	3.87%	5.01%	3.81%	-	-
pr124	11.82%	1.56%	5.91%	4.41%	**0.29%**	2.97%	1.22%	1.22%	0.82%	3.84%
bier127	20.65%	4.08%	3.76%	**1.71%**	60.56%	3.48%	3.79%	2.46%	2.40%	8.92%
ch130	16.53%	6.63%	2.85%	2.96%	**0.25%**	4.89%	5.48%	1.93%	1.06%	0.57%
pr136	9.14%	5.54%	5.62%	4.90%	**1.06%**	6.33%	5.14%	4.54%	1.74%	1.56%
pr144	21.30%	3.44%	1.28%	8.77%	**0.80%**	1.40%	3.44%	2.49%	4.56%	3.47%
ch150	21.26%	3.60%	3.08%	3.45%	**0.83%**	3.55%	3.45%	1.23%	-	-
KroA150	17.80%	6.93%	4.03%	9.98%	13.15%	4.51%	3.91%	3.91%	13.40%	**3.68%**
KroB150	20.20%	6.10%	5.52%	9.87%	11.72%	5.40%	4.10%	**2.82%**	7.80%	3.18%
pr152	16.20%	4.48%	2.92%	13.47%	4.11%	**2.17%**	3.59%	3.59%	2.20%	2.52%
u159	21.97%	6.84%	8.79%	7.38%	2.19%	7.67%	5.86%	3.16%	**1.51%**	10.84%
rat195	25.40%	6.93%	**2.84%**	16.57%	29.06%	9.90%	5.81%	4.99%	27.21%	10.81%
d198	13.83%	12.27%	**1.16%**	331.58%	45.98%	4.99%	10.74%	8.75%	-	-
KroA200	22.44%	3.60%	1.27%	15.64%	20.00%	7.01%	1.52%	**1.25%**	10.74%	6.14%
KroB200	23.69%	10.51%	**3.67%**	18.54%	21.06%	7.05%	6.28%	5.66%	-	-
Avg. Gap, [50,100)	6.53%	1.60%	2.93%	3.61%	4.88%	3.19%	0.59%	**0.46%**	4.42%	1.72%
Avg. Gap, [100,150)	9.69%	3.01%	3.29%	15.29%	8.16%	3.53%	2.74%	**1.57%**	5.20%	2.78%
Avg. Gap, [150,200]	12.76%	6.81%	3.70%	47.39%	16.45%	5.81%	5.03%	**3.93%**	10.48%	6.20%
Avg. Gap all	15.56%	3.90%	3.34%	22.83%	10.06%	4.17%	3.01%	**2.07%**	6.28%	3.34%

with a set of parameters θ defined as follows:

$$\pi_\theta(a|s) = Pr(A_t = a|S_t = s, \theta = \theta_t) \tag{22}$$

To find the optimal values of parameters θ, an objective function $J(\theta)$ needs to be defined, which is then maximized. Consequently, we need to solve the optimization problem with the gradient ascent to update the values of θ with step size α:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \tag{23}$$

For instance, the objective function can be set to maximize the total reward at the end of the episode, R_T then in accordance to the policy gradient theorem, we obtain:

$$\nabla_\theta J(\theta) = \mathbb{E}[\nabla_\theta \log \pi_\theta(a|s) R_T] \tag{24}$$

In fact, the REINFORCE algorithm widely adopted to solve VRPs, uses Monte-Carlo gradient methods to learn optimal policies. However, using the total reward of an episode results in high variance, which can be mitigated by the advantage function defined as the difference between the total reward and baseline. Accordingly, we can write the REINFORCE algorithm with baseline as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}[\nabla_\theta \log \pi_\theta(a|s)(R_T - b)] \tag{25}$$

In practice, actor-critic methods are often used, where an actor produces policy, and a critic is responsible for the policy evaluation. In contrast to Monte-Carlo gradient methods, TD is used to update values in actor-critic policy gradients.

REFERENCES

[1] M. A. Levans, "30th annual state of logistics report: What's next?" *Logistics Manage. Highlands Ranch*, vol. 58, no. 7, pp. 9–10, 2019.

[2] Y. Yuan, D. Cattaruzza, M. Ogier, and F. Semet, "Last mile delivery problem: The one-vehicle case," in *Proc. 7th Workshop Freight Transp. Logistics (Odysseus)*, Cagliari, Italy, Jun. 2018. [Online]. Available: https://hal.science/hal-01951934

[3] A. E. Rizzoli, R. Montemanni, E. Lucibello, and L. M. Gambardella, "Ant colony optimization for real-world vehicle routing problems," *Swarm Intell.*, vol. 1, no. 2, pp. 135–151, 2007.

[4] J. K. Lenstra and A. H. G. R. Kan, "Complexity of vehicle routing and scheduling problems," *Networks*, vol. 11, no. 2, pp. 221–227, Jun. 1981.

[5] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *Eur. J. Oper. Res.*, vol. 59, no. 3, pp. 345–358, Jun. 1992.

[6] K. F. Doerner and V. Schmid, "Survey: Matheuristics for rich vehicle routing problems," in *Proc. Int. Workshop Hybrid metaheuristics*. Cham, Switzerland: Springer, 2010, pp. 206–221.

[7] M. Asghari and S. M. J. M. Al-E-Hashem, "Green vehicle routing problem: A state-of-the-art review," *Int. J. Prod. Econ.*, vol. 231, Jan. 2021, Art. no. 107899.

[8] A. Bogyrbayeva, S. Jang, A. Shah, Y. J. Jang, and C. Kwon, "A reinforcement learning approach for rebalancing electric vehicle sharing systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 8704–8714, Jul. 2022.

[9] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 1–11.

[10] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: A methodological tour d'horizon," *Eur. J. Oper. Res.*, vol. 290, no. 2, pp. 405–421, Apr. 2021.

[11] N. Mazyavkina, S. Sviridov, S. Ivanov, and E. Burnaev, "Reinforcement learning for combinatorial optimization: A survey," *Comput. Oper. Res.*, vol. 134, Oct. 2021, Art. no. 105400.

[12] Y. Yan, A. H. F. Chow, C. P. Ho, Y.-H. Kuo, Q. Wu, and C. Ying, "Reinforcement learning for logistics and supply chain management: Methodologies, state of the art, and future opportunities," *Transp. Res. E, Logistics Transp. Rev.*, vol. 162, Jun. 2022, Art. no. 102712.

[13] N. Vesselinova, R. Steinert, D. F. Perez-Ramirez, and M. Boman, "Learning combinatorial optimization on graphs: A survey with applications to networking," *IEEE Access*, vol. 8, pp. 120388–120416, 2020.

[14] C. Zhang et al., "A review on learning to solve combinatorial optimisation problems in manufacturing," *IET Collaborative Intell. Manuf.*, vol. 5, no. 1, Mar. 2023, Art. no. e12072.

[15] Q. Cappart, D. Chételat, E. Khalil, A. Lodi, C. Morris, and P. Veličković, "Combinatorial optimization and reasoning with graph neural networks," 2021, *arXiv:2102.09544*.

[16] U. Junior Mele, L. Maria Gambardella, and R. Montemanni, "Machine learning approaches for the traveling salesman problem: A survey," in *Proc. 8th Int. Conf. Ind. Eng. Appl. (Europe)*, Jan. 2021, pp. 182–186.

[17] R. Bai et al., "Analytics and machine learning in vehicle routing research," *Int. J. Prod. Res.*, vol. 61, no. 1, pp. 1–27, Jan. 2023.

[18] B. Li, G. Wu, Y. He, M. Fan, and W. Pedrycz, "An overview and experimental study of learning-based optimization algorithms for the vehicle routing problem," *IEEE/CAA J. Autom. Sinica*, vol. 9, no. 7, pp. 1115–1138, Jul. 2022.

[19] M. Held and R. M. Karp, "A dynamic programming approach to sequencing problems," *J. Soc. Ind. Appl. Math.*, vol. 10, no. 1, pp. 196–210, Mar. 1962.

[20] M. Hahsler and K. Hornik, "TSP- infrastructure for the traveling salesperson problem," *J. Stat. Softw.*, vol. 23, no. 2, pp. 1–27, 2007.

[21] K. Helsgaun, "An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems," Roskilde Universitet, Tech. Rep., Dec. 2017.

[22] M. Prates, P. H. Avelar, H. Lemos, L. C. Lamb, and M. Y. Vardi, "Learning to solve Np-complete problems: A graph neural network for decision TSP," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 4731–4738.

[23] C. K. Joshi, T. Laurent, and X. Bresson, "An efficient graph convolutional network technique for the travelling salesman problem," 2019, *arXiv:1906.01227*.

[24] M. van Knippenberg, M. Holenderski, and V. Menkovski, "Complex vehicle routing with memory augmented neural networks," in *Proc. IEEE Conf. Ind. Cyberphysical Syst. (ICPS)*, vol. 1, Jun. 2020, pp. 303–308.

[25] A. Milan, S. H. Rezatofighi, R. Garg, A. Dick, and I. Reid, "Data-driven approximations to np-hard problems," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 1–7.

[26] C. K. Joshi, T. Laurent, and X. Bresson, "On learning paradigms for the travelling salesman problem," 2019, *arXiv:1910.07210*.

[27] C. K. Joshi, Q. Cappart, L.-M. Rousseau, and T. Laurent, "Learning the travelling salesman problem requires rethinking generalization," *Constraints*, vol. 27, no. 1, pp. 70–98, Apr. 2022, doi: 10.1007/s10601-022-09327-y.

[28] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–7.

[29] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[30] M. Riedmiller, "Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method," in *Proc. Eur. Conf. Mach. Learn.* Cham, Switzerland: Springer, 2005, pp. 317–328.

[31] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!" 2019, *arXiv:1803.08475*.

[32] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," 2016, *arXiv:1611.09940*.

[33] B. Peng, J. Wang, and Z. Zhang, "A deep reinforcement learning algorithm using dynamic attention model for vehicle routing problems," in *Proc. Int. Symp. Intell. Comput. Appl.* Cham, Switzerland: Springer, 2019, pp. 636–650.

[34] L. Xin, W. Song, Z. Cao, and J. Zhang, "Step-wise deep learning models for solving routing problems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 4861–4871, Jul. 2021.

[35] P. R. de O. da Costa, J. Rhuggenaath, Y. Zhang, and A. Akcay, "Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning," in *Proc. Asian Conf. Mach. Learn.*, 2020, pp. 465–480.

[36] A. Vaswani et al., "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[37] Y. Jiang, Y. Wu, Z. Cao, and J. Zhang, "Learning to solve routing problems via distributionally robust optimization," *Proc. AAAI Conf. Artif. Intell.*, Jun. 2022, vol. 36, no. 9, pp. 9786–9794.

[38] Y. Senuma, Z. Wang, Y. Nakano, and J. Ohya, "GEAR: A graph edge attention routing algorithm solving combinatorial optimization problem with graph edge cost," in *Proc. 10th ACM SIGSPATIAL Int. Workshop Analytics Big Geospatial Data*, 2022, pp. 8–16.

[39] K. Li, T. Zhang, R. Wang, Y. Wang, Y. Han, and L. Wang, "Deep reinforcement learning for combinatorial optimization: Covering salesman problems," *IEEE Trans. Cybern.*, vol. 52, no. 12, pp. 13142–13155, Dec. 2022.

[40] K. Zhang, F. He, Z. Zhang, X. Lin, and M. Li, "Multi-vehicle routing problems with soft time windows: A multi-agent reinforcement learning approach," *Transp. Res. C, Emerg. Technol.*, vol. 121, Dec. 2020, Art. no. 102861.

[41] L. Xin, W. Song, Z. Cao, and J. Zhang, "Multi-decoder attention model with embedding glimpse for solving vehicle routing problems," in *Proc. 35th AAAI Conf. Artif. Intell.*, 2021, pp. 12042–12049.

[42] M. Kim et al., "Learning collaborative policies to solve NP-hard routing problems," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 10418–10430.

[43] Y.-D. Kwon, J. Choo, B. Kim, I. Yoon, Y. Gwon, and S. Min, "POMO: Policy optimization with multiple optima for reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, vol. 33, 2020, pp. 21188–21198.

[44] Y. Liang Goh, W. Sun Lee, X. Bresson, T. Laurent, and N. Lim, "Combining reinforcement learning and optimal transport for the traveling salesman problem," 2022, *arXiv:2203.00903*.

[45] P. Emami and S. Ranka, "Learning permutations with sinkhorn policy gradient," 2018, *arXiv:1805.07010*.

[46] J. J. Q. Yu, W. Yu, and J. Gu, "Online vehicle routing with neural combinatorial optimization and deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 10, pp. 3806–3817, Oct. 2019.

[47] B. Lin, B. Ghaddar, and J. Nathwani, "Deep reinforcement learning for the electric vehicle routing problem with time windows," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 11528–11538, Aug. 2022.

[48] W. Pan and S. Q. Liu, "Deep reinforcement learning for the dynamic and uncertain vehicle routing problem," *Appl. Intell.*, vol. 53, no. 1, pp. 405–422, 2023.

[49] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR), Conf. Track*, Y. Bengio and Y. LeCun, Eds. San Diego, CA, USA, May 2015. [Online]. Available: http://arxiv.org/abs/1409.0473

[50] J. M. Vera and A. G. Abad, "Deep reinforcement learning for routing a heterogeneous fleet of vehicles," in *Proc. IEEE Latin Amer. Conf. Comput. Intell. (LA-CCI)*, 2019, pp. 1–6.

[51] J. K. Falkner and L. Schmidt-Thieme, "Learning to solve vehicle routing problems with time windows through joint attention," 2020, *arXiv:2006.09100*.

[52] K. Lei, P. Guo, Y. Wang, X. Wu, and W. Zhao, "Solve routing problems with a residual edge-graph attention neural network," *Neurocomputing*, vol. 508, pp. 79–98, Oct. 2022.

[53] U. Gunarathna, R. Borovica-Gajic, S. Karunasekara, and E. Tanin, "Solving dynamic graph problems with multi-attention deep reinforcement learning," 2022, *arXiv:2201.04895*.

[54] J. Li et al., "Learning feature embedding refiner for solving vehicle routing problems," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jun. 23, 2023, doi: 10.1109/TNNLS.2023.3285077.

[55] K. Li, T. Zhang, and R. Wang, "Deep reinforcement learning for multiobjective optimization," *IEEE Trans. Cybern.*, vol. 51, no. 6, pp. 3103–3114, Jun. 2021.

[56] A. Hottung, Y.-D. Kwon, and K. Tierney, "Efficient active search for combinatorial optimization problems," 2021, *arXiv:2106.05126*.

[57] Q. Sykora, M. Ren, and R. Urtasun, "Multi-agent routing value iteration network," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 9300–9310.

[58] Y. Xu, M. Fang, L. Chen, G. Xu, Y. Du, and C. Zhang, "Reinforcement learning with multiple relational attention for solving vehicle routing problems," *IEEE Trans. Cybern.*, vol. 52, no. 10, pp. 11107–11120, Oct. 2022.

[59] I. Drori et al., "Learning to solve combinatorial optimization problems on real-world graphs in linear time," in *Proc. 19th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2020, pp. 19–24.

[60] J. Li, L. Xin, Z. Cao, A. Lim, W. Song, and J. Zhang, "Heterogeneous attentions for solving pickup and delivery problem via deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 2306–2315, Mar. 2022.

[61] J. Li et al., "Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem," *IEEE Trans. Cybern.*, vol. 52, no. 12, pp. 13572–13585, Dec. 2022.

[62] L. Duan et al., "Efficiently solving the practical vehicle routing problem: A novel joint learning approach," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2020, pp. 3054–3063.

[63] F. Alesiani, "BiGrad: Differentiating through bilevel optimization programming," in *Proc. AAAI Workshop Adversarial Mach. Learn. Beyond*, 2022, pp. 1–11. [Online]. Available: https://openreview.net/forum?id=HvRAM-dpmEv

[64] M. V. Pogancic, A. Paulus, V. Musil, G. Martius, and M. Rolínek, "Differentiation of blackbox combinatorial solvers," in *Proc. 8th Int. Conf. Learn. Represent. (ICLR)*. Addis Ababa, Ethiopia. OpenReview.net, Apr. 2020. [Online]. Available: https://openreview.net/forum?id=BkevoJSYPB

[65] L. M. Gambardella and M. Dorigo, "ANT-Q: A reinforcement learning approach to the traveling salesman problem," in *Proc. Mach. Learn.*, 1995, pp. 252–260.

[66] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.

[67] F. Liu and G. Zeng, "Study of genetic algorithm with reinforcement learning to solve the TSP," *Exp. Syst. Appl.*, vol. 36, no. 3, pp. 6995–7001, Apr. 2009.

[68] M. M. Alipour, S. N. Razavi, M. R. Feizi Derakhshi, and M. A. Balafar, "A hybrid algorithm using a genetic algorithm and multiagent reinforcement learning heuristic to solve the traveling salesman problem," *Neural Comput. Appl.*, vol. 30, no. 9, pp. 2935–2951, Nov. 2018.

[69] 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, Aug. 2022.

[70] Y. Niu, J. Shao, J. Xiao, W. Song, and Z. Cao, "Multi-objective evolutionary algorithm based on RBF network for solving the stochastic vehicle routing problem," *Inf. Sci.*, vol. 609, pp. 387–410, Sep. 2022.

[71] A. Hottung and K. Tierney, "Neural large neighborhood search for the capacitated vehicle routing problem," 2019, *arXiv:1911.09539*.

[72] A. A. Syed, K. Akhnoukh, B. Kaltenhaeuser, and K. Bogenberger, "Neural network based large neighborhood search algorithm for ride hailing services," in *Proc. EPIA Conf. Artif. Intell.* Cham, Switzerland: Springer, 2019, pp. 584–595.

[73] M. A. L. Silva, S. R. de Souza, M. J. F. Souza, and A. L. C. Bazzan, "A reinforcement learning-based multi-agent framework applied for solving routing and scheduling problems," *Exp. Syst. Appl.*, vol. 131, pp. 148–171, Oct. 2019.

[74] F. C. Fernandes, S. R. de Souza, M. A. L. Silva, H. E. Borges, and F. F. Ribeiro, "A multiagent architecture for solving combinatorial optimization problems through metaheuristics," in *Proc. IEEE Int. Conf. Syst., Man Cybern.*, Oct. 2009, pp. 3071–3076.

[75] M. W. Ulmer, J. C. Goodson, D. C. Mattfeld, and M. Hennig, "Offline–online approximate dynamic programming for dynamic vehicle routing with stochastic requests," *Transp. Sci.*, vol. 53, no. 1, pp. 185–202, Feb. 2019.

[76] M. W. Ulmer, D. C. Mattfeld, and F. Köster, "Budgeting time for dynamic vehicle routing with stochastic customer requests," *Transp. Sci.*, vol. 52, no. 1, pp. 20–37, Jan. 2018.

[77] Q. Cappart, T. Moisan, L.-M. Rousseau, I. Prémont-Schwarz, and A. A. Cire, "Combining reinforcement learning and constraint programming for combinatorial optimization," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 5, 2021, pp. 3677–3687.

[78] W. Kool, H. van Hoof, J. Gromicho, and M. Welling, "Deep policy dynamic programming for vehicle routing problems," 2021, *arXiv:2102.11756*.

[79] S. Xu, S. S. Panwar, M. Kodialam, and T. Lakshman, "Deep neural network approximated dynamic programming for combinatorial optimization," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 2, 2020, pp. 1684–1691.

[80] F. Yang, T. Jin, T.-Y. Liu, X. Sun, and J. Zhang, "Boosting dynamic programming with neural networks for solving Np-hard problems," in *Proc. Asian Conf. Mach. Learn.*, 2018, pp. 726–739.

[81] A. Delarue, R. Anderson, and C. Tjandraatmadja, "Reinforcement learning with combinatorial actions: An application to vehicle routing," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 609–620.

[82] R. Anderson, J. Huchette, W. Ma, C. Tjandraatmadja, and J. P. Vielma, "Strong mixed-integer programming formulations for trained neural networks," *Math. Program.*, vol. 183, nos. 1–2, pp. 3–39, Sep. 2020.

[83] F. Alesiani, G. Ermis, and K. Gkiotsalitis, "Constrained clustering for the capacitated vehicle routing problem (CC-CVRP)," *Appl. Artif. Intell.*, vol. 36, no. 1, Dec. 2022, Art. no. 1995658.

[84] Y. Wang, L. Ran, X. Guan, J. Fan, Y. Sun, and H. Wang, "Collaborative multicenter vehicle routing problem with time windows and mixed deliveries and pickups," *Exp. Syst. Appl.*, vol. 197, Jul. 2022, Art. no. 116690.

[85] S.-H. Lu, R. J. Kuo, Y.-T. Ho, and A.-T. Nguyen, "Improving the efficiency of last-mile delivery with the flexible drones traveling salesman problem," *Exp. Syst. Appl.*, vol. 209, Dec. 2022, Art. no. 118351.

[86] Y. Wang et al., "Collaborative two-echelon multicenter vehicle routing optimization based on state–space–time network representation," *J. Cleaner Prod.*, vol. 258, Jun. 2020, Art. no. 120590.

[87] Y. Wang, S. Peng, X. Zhou, M. Mahmoudi, and L. Zhen, "Green logistics location-routing problem with eco-packages," *Transp. Res. E, Logistics Transp. Rev.*, vol. 143, Nov. 2020, Art. no. 102118.

[88] Y. Ma et al., "Learning to iteratively solve routing problems with dual-aspect collaborative transformer," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 11096–11107.

[89] Y. Wu, W. Song, Z. Cao, J. Zhang, and A. Lim, "Learning improvement heuristics for solving routing problems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 9, pp. 5057–5069, Sep. 2022.

[90] X. Chen and Y. Tian, "Learning to perform local rewriting for combinatorial optimization," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8–14, 2019, Vancouver, BC, Canada*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 6278–6289. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/131f383b434fdf48079bff1e44e2d9a5-Abstract.html

[91] L. Gao, M. Chen, Q. Chen, G. Luo, N. Zhu, and Z. Liu, "Learn to design the heuristics for vehicle routing problem," 2020, *arXiv:2002.08539*.

[92] S. Li, Z. Yan, and C. Wu, "Learning to delegate for large-scale vehicle routing," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 26198–26211.

[93] M. Deudon, P. Cournut, A. Lacoste, Y. Adulyasak, and L.-M. Rousseau, "Learning heuristics for the TSP by policy gradient," in *Proc. Int. Conf. Integr. Constraint Program., Artif. Intell., Oper. Res.* Cham, Switzerland: Springer, 2018, pp. 170–181.

[94] J. Zhao, M. Mao, X. Zhao, and J. Zou, "A hybrid of deep reinforcement learning and local search for the vehicle routing problems," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 11, pp. 7208–7218, Nov. 2021.

[95] Q. Ma, S. Ge, D. He, D. Thaker, and I. Drori, "Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning," 2019, *arXiv:1911.04936*.

[96] R. Zhang, A. Prokhorchuk, and J. Dauwels, "Deep reinforcement learning for traveling salesman problem with time windows and rejections," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–8.

[97] Y. Ma et al., "Efficient neural neighborhood search for pickup and delivery problems," 2022, *arXiv:2204.11399*.

[98] K. Lin, R. Zhao, Z. Xu, and J. Zhou, "Efficient large-scale fleet management via multi-agent deep reinforcement learning," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 1774–1783.

[99] Z.-H. Fu, K.-B. Qiu, and H. Zha, "Generalize a small pre-trained model to arbitrarily large TSP instances," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 8, pp. 7474–7482.

[100] Z. Xing and S. Tu, "A graph neural network assisted Monte Carlo tree search approach to traveling salesman problem," *IEEE Access*, vol. 8, pp. 108418–108428, 2020.

[101] J. Ragan-Kelley, C. Barnes, A. Adams, S. Paris, F. Durand, and S. Amarasinghe, "Halide: A language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines," *ACM SIGPLAN Notices*, vol. 48, no. 6, pp. 519–530, 2013.

[102] X. Chen, M. W. Ulmer, and B. W. Thomas, "Deep Q-learning for same-day delivery with vehicles and drones," *Eur. J. Oper. Res.*, vol. 298, no. 3, pp. 939–952, May 2022.

[103] R. Tyasnurita, E. Özcan, and R. John, "Learning heuristic selection using a time delay neural network for open vehicle routing," *Proc. IEEE Congr. Evol. Comput. (CEC)*, Oct. 2017, pp. 1474–1481.

[104] A. Hottung, B. Bhandari, and K. Tierney, "Learning a latent search space for routing problems using variational autoencoders," in *Proc. 9th Int. Conf. Learn. Represent. (ICLR)*. Virtual Event, Austria: OpenReview.net, May 2021. [Online]. Available: https: //openreview.net/forum?id=90JprVrJBO

[105] A. E. Gutierrez-Rodríguez, S. E. Conant-Pablos, J. C. Ortiz-Bayliss, and H. Terashima-Marín, "Selecting meta-heuristics for solving vehicle routing problems with time windows via meta-learning," *Exp. Syst. Appl.*, vol. 118, pp. 470–481, Mar. 2019.

[106] K. C. Tan, Y. H. Chew, and L. H. Lee, "A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows," *Comput. Optim. Appl.*, vol. 34, no. 1, pp. 115–151, May 2006.

[107] B. Ombuki, B. J. Ross, and F. Hanshar, "Multi-objective genetic algorithms for vehicle routing problem with time windows," *Appl. Intell.*, vol. 24, no. 1, pp. 17–30, 2006.

[108] Y. Qi, Z. Hou, H. Li, J. Huang, and X. Li, "A decomposition based memetic algorithm for multi-objective vehicle routing problem with time windows," *Comput. Oper. Res.*, vol. 62, pp. 61–77, 2015.

[109] D. Q. Wu, M. Dong, and H. Y. Li, "Vehicle routing problem with time windows using multi-objective co-evolutionary approach," *Int. J. Simul. Model.*, vol. 15, no. 4, pp. 742–753, Dec. 2016.

[110] Z.-H. Fu, K.-B. Qiu, M. Qiu, and H. Zha, "Targeted sampling of enlarged neighborhood via Monte Carlo tree search for TSP," in *Proc. ICLR*, 2019, pp. 1–11.

[111] L. Xin, W. Song, Z. Cao, and J. Zhang, "NeuroLKH: Combining deep learning model with lin-kernighan-helsgaun heuristic for solving the traveling salesman problem," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021.

[112] W. Joe and H. C. Lau, "Deep reinforcement learning approach to solve dynamic vehicle routing problem with stochastic customers," in *Proc. Int. Conf. Automated Planning Scheduling*, vol. 30, 2020, pp. 394–402.

[113] Z. Zong, H. Wang, J. Wang, M. Zheng, and Y. Li, "RBG: Hierarchically solving large-scale routing problems in logistic systems via reinforcement learning," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, 2022, pp. 4648–4658.

[114] J. Zheng, K. He, J. Zhou, Y. Jin, and C.-M. Li, "Reinforced lin–kernighan–helsgaun algorithms for the traveling salesman problems," *Knowl.-Based Syst.*, vol. 260, Jan. 2023, Art. no. 110144.

[115] Z. Sun, M. Greiff, A. Robertsson, and R. Johansson, "Feasible coordination of multiple homogeneous or heterogeneous mobile vehicles with various constraints," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 1008–1013.

[116] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 30, no. 1, 2016.

[117] W. Qin, Z. Zhuang, Z. Huang, and H. Huang, "A novel reinforcement learning-based hyper-heuristic for heterogeneous vehicle routing problem," *Comput. Ind. Eng.*, vol. 156, Jun. 2021, Art. no. 107252.

[118] A. Bogyrbayeva, T. Yoon, H. Ko, S. Lim, H. Yun, and C. Kwon, "A deep reinforcement learning approach for solving the traveling salesman problem with drone," *Transp. Res. C, Emerg. Technol.*, vol. 148, p. 103981, 2023.

[119] G. Bono, J. S. Dibangoye, O. Simonin, L. Matignon, and F. Pereyron, "Solving multi-agent routing problems using deep attention mechanisms," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 12, pp. 7804–7813, Dec. 2021.

[120] J. Banfi, A. Messing, C. Kroninger, E. Stump, S. Hutchinson, and N. Roy, "Hierarchical planning for heterogeneous multi-robot routing problems via learned subteam performance," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 4464–4471, Apr. 2022.

[121] P. da Costa, J. Rhuggenaath, Y. Zhang, A. Akcay, and U. Kaymak, "Learning 2-Opt heuristics for routing problems via deep reinforcement learning," *Social Netw. Comput. Sci.*, vol. 2, no. 5, pp. 1–6, Sep. 2021.

[122] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Oper. Res.*, vol. 35, no. 2, pp. 254–265, Apr. 1987.

[123] L. Perron and V. Furnon. (2013). OR-Tools. Google. [Online]. Available: https://developers.google.com/optimization/

[124] T. Jacobs, F. Alesiani, and G. Ermis, "Reinforcement learning for route optimization with robustness guarantees," in *Proc. IJCAI*, 2021, pp. 2592–2598.

[125] Y. Jiang, Z. Cao, Y. Wu, and J. Zhang, "Multi-view graph contrastive learning for solving vehicle routing problems," in *Proc. Uncertainty Artif. Intell.*, 2023, pp. 984–994.

[126] J. Bi et al., "Learning generalizable models for vehicle routing problems via knowledge distillation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 31226–31238.

[127] J. Zhou, Y. Wu, W. Song, Z. Cao, and J. Zhang, "Towards omni-generalizable neural methods for vehicle routing problems," 2023, *arXiv:2305.19587*.

[128] L. Accorsi, A. Lodi, and D. Vigo, "Guidelines for the computational testing of machine learning approaches to vehicle routing problems," *Oper. Res. Lett.*, vol. 50, no. 2, pp. 229–234, Mar. 2022.

[129] Á. Corberán, R. Eglese, G. Hasle, I. Plana, and J. M. Sanchis, "Arc routing problems: A review of the past, present, and future," *Networks*, vol. 77, no. 1, pp. 88–115, Jan. 2021.