Learning to Solve Vehicle Routing Problems: A Survey

Aigerim Bogyrbayeva†, Meraryslan Meraliyev ♭, Taukekhan Mustakhov†, 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 a great interest from both machine learning and operations research communities to solve VRPs either by pure learning methods or by combining them with the traditional hand-crafted heuristics. We present the taxonomy of the studies for learning paradigms, solution structures, underlying models, and algorithms. We present in detail the results of the state-of-the-art methods demonstrating their competitiveness with the traditional methods. The paper outlines the future research directions to incorporate learning-based solutions to overcome the challenges of modern transportation systems.**

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

I. Introduction

Cost-effective logistics systems overall define the competitiveness of the companies, and the relation of logistics expenditure to GDP indicates the effectiveness of business operations in a country. For instance, in 2018, the 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, the transportation costs are mainly influenced by the last-mile delivery, which is 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 substitutes for 50% of the total transportation costs [2]. Also, the carbon dioxide footprint is another emerging concern in logistics systems. In order 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, are defined in 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 a 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 the minimal total cost for M number of identical vehicles leaving from a depot, v_0, to visit all the nodes,$\mathbb{V}\backslash v_0$, representing customers with non-negative demand d_v, and return to the depot, where each customer is visited only once.

†Department of Computer Science, Suleyman Demirel University (SDU), Qaskelen, Kazakhstan Email: {aigerim.bogyrbayeva, taukekhan.mustakhov, b.dauletbayev}@sdu.edu.kz

♭ corresponding author. Department of Computer Science, Suleyman Demirel University (SDU), Qaskelen, Kazakhstan Email: meraryslan.meraliyev@sdu.edu.kz

This work is supported by the Computer Science Department funding and the internal grant of SDU for 2022-2023.

Different constraints are added to VRP to reflect the realistic routing challenges faced by delivery companies [3]. In Capacitated Vehicle Routing Problem (CVRP), each vehicle is subjected to 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 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).

As generalizations of Travelling Salesman Problem (TSP), VRPs belong to the family of combinatorial optimization 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 hand-crafted heuristics are built on the experts' domain knowledge about the specifics of the structure of the problem at hand. 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* approximately functions instead of explicitly writing computer programs. The main advantage of learning models is their ability to generalize to input problems coming from identical distributions and produce solutions instantaneously [9]. 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. The figure 1 indicates the increasing trend in the number of such studies.

There are two main learning paradigms used for solving VRPs. In Supervised Learning, a model is trained on a training dataset consisting of input problems and corresponding solutions. In Reinforcement Learning, 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, which led to new methods that combine the traditional heuristics with learning methods. An increasingly growing literature has attracted the attention of both machine learning and operations research communities [10].

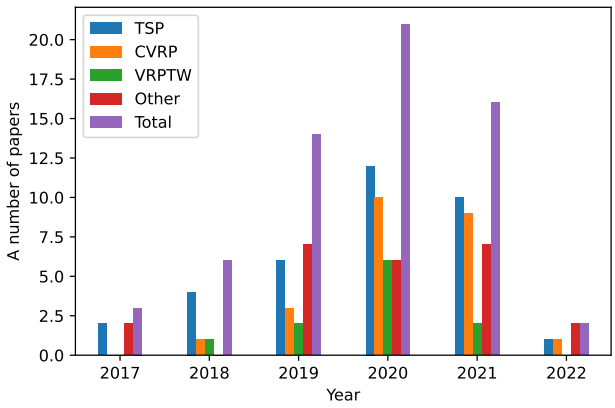

Fig. 1: A number of studies focused on solving VRPs using learning methods.

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.

Compared to the existing surveys [11, 12, 13, 14], the presented study does not limit its scope to Reinforcement Learning methods only but surveys all studies that used any learning paradigm. Also, we focus our attention on routing problems in particular rather than discussing the combinatorial optimization problems in general. We consider our work to be complementary to the existing studies.

Our contributions can be summarized as follows:

- This is the first survey focused on learning methods to solve routing problems in general.
- We present taxonomy to classify the main research directions in the existing literature that includes both Supervised and Reinforcement Learning Methods.
- We provide detailed overview of the hybrid models that combine the hand-crafted heuristics with learning methods.

We have the following outline for the paper: in Section II, we provide an overview of learning paradigms, models, and algorithms; in Section III, we present the central taxonomy for learning methods to solve routing problems and discuss end-to-end learning methods; in Section IV, we focus on hybrid methods and present different solution structures; in Section V we present single and multiple VRP formulations and results; in Section VI, we discuss the future research directions and finalize with the concluding remarks.

II. Background

This section discusses two learning paradigms applied to solve VRPs, namely Supervised Learning and Reinforcement Learning. Unlike the former, Reinforcement Learning does not rely on given labels but instead focuses on learning through interactions in order to maximize a long-term reward. A goal-oriented reinforcement learning *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 reinforcement learning algorithms that solve sequential decision-making problems.

A. Supervised Learning

In supervised learning, 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. Then we can assemble a training set $D = (x_i, y_i), i = 1, \ldots, N$ and $x_i \in \mathbb{R}^p$. A supervised learning 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. For the recent survey of the loss function, please see [15]. 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, reinforcement learning 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 reward in the next step denoted by R_{t+1} and a new state S_{t+1}. The state-transition probability function given in 1 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:

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

The reward function determines the expected reward for the taken action a at state s with the consecutive state s':

$$r(s,a,s') = \mathbb{E}[R_{t+1}|S_t = s, A_t = a, S_{t+1} = s']. \tag{2}$$

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

$$\pi(a|s) = p(A_{t+1} = a|S_t = s). \tag{3}$$

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

$$v_\pi(s) = \mathbb{E}[\sum_{k=0}^{T-1} \gamma^k R_{t+k+1}|S_t = s] \tag{4}$$

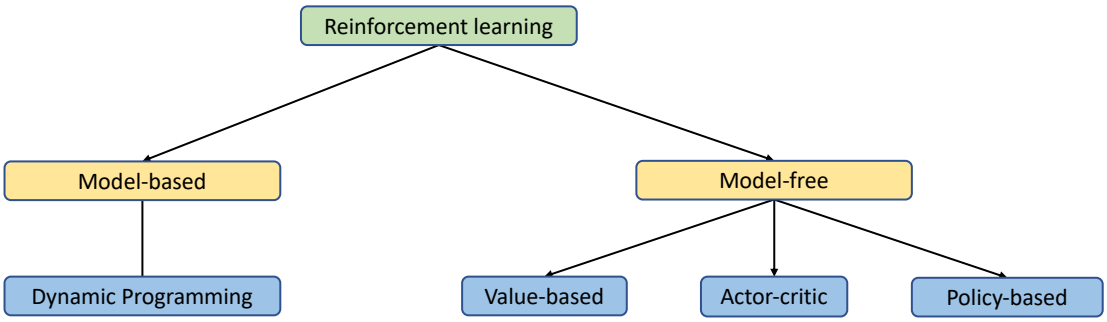

Fig. 2: An overview of reinforcement learning methods.

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

$$q_\pi(s,a) = \mathbb{E}[\sum_{k=0}^{T-1} \gamma^k R_{t+k+1} | S_t = s, A_t = a.] \quad (5)$$

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 $v^*(s)$:

$$v^*(s) = \max_\pi v_\pi(s) = \max_a \mathbb{E}[R_{t+1} + \gamma v^*(S_{t+1}) | S_t = s, A_t = a] \quad (6)$$

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

$$q^*(s,a) = \max_\pi q_\pi(s,a) = \max_{a'} \mathbb{E}[R_{t+1} + \gamma q^*(S_{t+1}, a') | S_t = s, A_t = a] \quad (7)$$

Equations (6) - (7) are known as the Bellman optimality equations that are recursive.

C. Reinforcement Learning Algorithms

We can broadly define *reinforcement learning* as any method that solves the above-defined MDPs. When Equations (2) and (3) 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 is one of such methods. Often Equations (2) and (3) are unknown, and one needs to learn the dynamics of the problem through experiences. 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 2 provides an overview of reinforcement learning. 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, we aim to find the action-value function, q^*. One of the widely used algorithms for such purpose is *Q-learning* [16] that recursively updates the values of action and state pairs independently from a policy with α as a constant step size:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \quad (8)$$

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 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 with a set of parameters θ defined as follows:

$$\pi_\theta(a|s) = Pr(A_t = a | S_t = s, \theta = \theta_t) \quad (9)$$

To find the optimal values of parameters θ, we need to define an objective function $J(\theta)$, which we aim to maximize. Then we need to solve the optimization problem with the gradient ascent to update the values of θ with step size α:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \quad (10)$$

For instance, the objective function can be set to maximize the total reward at the end of the episode, R_T then according to the policy gradient theorem [17] we have:

$$\nabla_\theta J(\theta) = \mathbb{E}[\nabla_\theta \log \pi_\theta(a|s) R_T] \quad (11)$$

In fact, the REINFORCE algorithm [18] widely adopted to solve VRPs uses Monte-Carlo gradient methods as in Equation (12) 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:

$$\nabla_\theta J(\theta) = \mathbb{E}[\nabla_\theta \log \pi_\theta(a|s)(R_T - b)] \quad (12)$$

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.

In recent years neural networks that, according to the universal approximation theorem, can approximate any continuous function with mild assumptions [19, 20] have been combined with the reinforcement learning algorithms mentioned above that gave rise to *Deep Reinforcement Learning* (DRL). For instance, in DRL, two separate neural networks are trained to estimate policy and state-value functions according to actor-critic methods [21, 22, 23]. Similarly, new algorithms have been proposed to use neural networks for Q-learning with high-dimensional data [24, 25, 26].

D. Multi-agent Reinforcement Learning

The algorithms, as mentioned earlier, focus on finding optimal policies for a single agent. However, real-life applications such as VRPs require learning policies for a fleet of vehicles. In general, sequential decision problems that involve several agents that either are fully competing, fully collaborating, or with mixed strategies can be formulated with Multi-agent Reinforcement Learning (MARL). Unlike single-agent reinforcement learning, in MARL, due to the presence of several agents in a shared environment, the stationary assumption for MDPs is violated [27]. Therefore, recently developed deep MARL algorithms [28, 29, 30] use Partially Observable MDP, which allows for each agent to have a local observation of the environment. Another challenge of MARL is scalability coming from increased complexity with the number of agents [31]. Along with the theoretical challenges of MARL, communication between agents becomes essential in the setting of many applications that may require training additional models to pass messages between agents [32].

III. Taxonomy of Learning Methods for Routing Problems

No single classification will cover the wide range of learning methods applied to solve routing problems. However, one broad classification can be based on the structure of the solutions, where only learning methods are used to solve routing problems or combined with existing non-learning methods. In *end-to-end* learning methods, either supervised or reinforcement learning 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 later will be further improved with construction heuristics or learning methods facilitate to solve the inner-problems of the existing non-learning methods to solve routing problems.

Another broad classification can be based on the routing problem types involving single or multiple vehicles. In principle, one can formulate the problems as single-agent reinforcement learning or multi-agent reinforcement learning problems. However, due to the shared action space among multiple vehicles, a central-controller is often used to route a fleet of heterogeneous or homogeneous vehicles. Figure 3 summarizes the taxonomy of learning methods to solve VRPs. Also, since the majority of the studies focus on routing a single vehicle, we refer to the underlying problems as TSP, CVRP, VRPTW and use mTSP, mCVRP, mVRPTW to indicate multiple vehicles.

A. End-to-end Supervised Learning for VRPs

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

In the case of TSP, to produce non-autoregressive solution, Held-Calp [33] algorithm or well-known solvers such as Concorde [34], LKH3 [35] is used to produce 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 into the adjacency matrix, where for each $e_{i,j} \in \mathbb{E}$ we can compute the probability $p_{i,j}$:

$$p_{i,j} = \begin{cases} 1, & \text{if } e_{i,j} \text{ is in } \pi^* \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

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

$$\mathcal{L}(\theta) = p_{i,j} \log \hat{p}_{i,j} - (1 - p_{i,j}) \log(1 - \hat{p}_{i,j}) \tag{14}$$

The adjacency matrix produced by the model can either follow *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.

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 autoregressive supervised learning methods:

$$\theta = \operatorname{argmax} \log p(\pi|\pi^*, \theta) \tag{15}$$

$$p(\pi|\pi^*, \theta) = \prod_{i=1}^{n} p(v_i|v_0, \ldots, v_{i-1}, \pi^*, \theta). \tag{16}$$

Both methods utilize advanced deep learning models such as Graph Neural Networks (GNNs) [36] and Graph Convolution Networks [37] to extract features of a graph and deploy Memory Augmented Neural Networks [38] and Recurrent Neural Networks (RNNs) [39] to pass sequential information. Both methods require training separate sets of parameters for different graph sizes to produce near-optimal solutions for TSP [40, 41]. Table I summarizes studies focused on end-to-end supervised learning. However, for more complex VRPs, the supervised learning method is coupled with non-learning methods, or it is fully replaced by reinforcement learning, which a priori does not require known solutions to the problems.

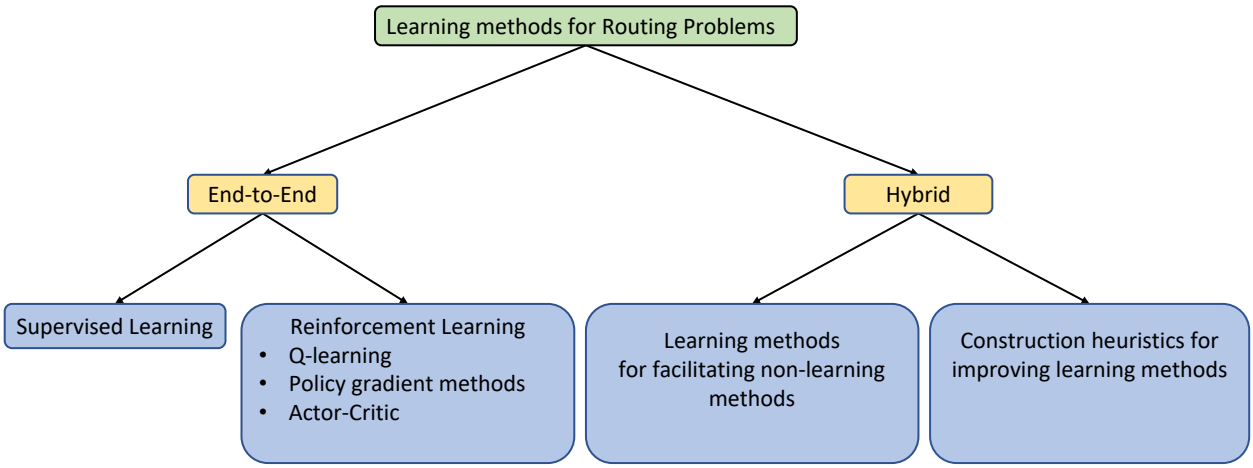

Fig. 3: The taxonomy of learning methods for VRPs.

TABLE I: The summary of supervised learning methods with autoregressive (AR) and None-autoregressive (NAR) solutions

Name	Problems	AR	NAR	Loss	Label
Joshi et al. [37]	TSP		✓	Cross-entropy	Concorde
Milan et al. [39]	TSP	✓		Log-likelihood	Nearest Neighbours
Prates et al. [36]	a decision TSP			Binary cross-entropy	Concorde
Joshi et al. [40]	TSP	✓		Cross-entropy	Concorde
Joshi et al. [41]	TSP	✓	✓	Cross-entropy	Concorde
Van Knippenberg et al. [38]	CVRP, TSP	✓		Cross-entropy	Held-Kalp algorithm, LKH-3, Concorde

B. End-to-end Deep Reinforcement Learning for VRPs

While *tabular* reinforcement learning methods are mostly used in the hybrid setting, in end-to-end methods, deep reinforcement learning is a popular choice due to the combinatorial nature of the action space of VRPs. We can categorize deep reinforcement learning 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 Q-function for each state and action pair from which an optimal policy is derived. However, instead of using one-step updates as in Equation (17), 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 $R_{t,t+n} = \sum_{k=0}^{n} r_{t+k}$:

$$\hat{Q}_\theta(S_t, A_t) \leftarrow \hat{Q}_\theta(S_t, A_t) + \\ + \alpha[R_{t,t+n} + \gamma \max_a \hat{Q}_\theta(S_{t+n}, a_{t+n}) - \hat{Q}_\theta(S_t, A_t)]. \tag{17}$$

For instance, [42] points out that using experience replay results in fast convergence to solve combinatorial optimization problems, which is a general case when neural networks are used as approximation functions [24, 43].

Even though value-based models are generally sample-efficient as compared to the 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 chain rule:

$$\pi_\theta(a|s) = \prod_{t=0}^{T} p(a_{t+1}|s_t, Y_t) \tag{18}$$

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:

$$\theta_c \leftarrow \theta_c + \alpha[\frac{1}{B}\sum_{j=1}^{B} \nabla_{\theta_c}(R^j - \hat{V}^j_{\theta_c}(S^j_0))^2] \tag{19}$$

Accordingly, the actor parameters are updated using REINFORCE algorithm:

$$\theta_a \leftarrow \theta_a + \alpha[\frac{1}{B}\sum_{j=1}^{B} \nabla_{\theta_a} \log \pi_{\theta_a}(R^j - \hat{V}^j_{\theta_c}(S_0))] \tag{20}$$

Some studies [44] use greedy rollout of the actor with the current set of parameters to avoid the complexities of training the critic network for Equation (19).

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 [45]. 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:

$$h_v^s = F^s(v_s), \quad h_d^v = F^d(v_d) \quad \forall v \in \mathbb{V}. \tag{21}$$

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:

$$h_v^{d_t} = F^d(v_{d_t}) \quad \forall v \in \mathbb{V}. \tag{22}$$

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:

$$\hat{h}_t = \frac{1}{|\mathbb{V}|} \sum_{v \in \mathbb{V}} h_v^t \tag{23}$$

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. [46, 47] has shown the value of dynamic encoders in improving the results to solve VRPs.

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 reinforcement learning algorithms used, VRPs require to learn 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, [45] 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, [48] proposed a novel graph representation called Structure2Vec that can encode both the graph and the partial solution at any time step. [44] proposes fully attention-based encoder introducing transformers [49] to solve VRPs, while [41] uses GNNs, a deep learning model dedicated to learn graph information. In return, other studies adopt the proposed encoders including Pointer Networks [50], multi-head attention [51, 52, 53, 54], recurrent neural networks (RNNs) [55], Structure2vec [56, 57] 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:

$$u_t = F(\hat{h}_t, F(s_t)), \quad \pi_\theta(a|s) = \text{softmax}(u_t) \tag{24}$$

where, F can be a series of nonlinear transformations. RNNs, which enables to pass information about the sequence of the partial solution, have been a common choice for a decoder that is followed by additive attention [58] to compute the alignment score between the graph embedding and the current state [9, 45, 59]. 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 [44, 46, 52, 60]. Table II summarizes studies with end-to-end reinforcement learning methods.

However, the greedy rollout resulting from a trained end-to-end reinforcement learning method may not be competitive against the hand-crafted 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 a specifics of an instance, thus improving the solution quality [61]. Another search method is *random sampling*, whereby indicating the number of total solutions to be sampled, among them, we select the best solution for a reward.

IV. Hybrid Methods

Hybrid Methods aim to take advantage of both learning and non-learning methods to solve challenging VRPs. Based on the structure of the hybrid method solutions, we can categorize them into two groups. In the first group, learning methods serve as a supporting tool to address the internal subproblems of non-learning methods. In the second group, learning methods are a primary tool to produce a solution, which is improved with construction heuristics.

A. Learning Methods for Facilitating non-learning methods

Combining Q-learning with meta-heuristics dates back to [70], which proposed to use ant-colony in a distributed setting 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, [71] 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.

[72] combines the genetic algorithm with reinforcement learning 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 the local search. Similarly, [73] combines the genetic algorithm with multi-agent reinforcement learning, where MARL is used to generate an initial solution which later will be improved by a genetic algorithm.

TABLE II: The summary of studies with end-to-end reinforcement learning 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

Study	Problems	Training Algorithm	Graph Embedding	Decoder
Bello et al. [45]	TSP	REINFORCE with baseline	Pointer Network	Pointer Network
James et al. [56]	DVRP	REINFORCE with baseline	Structure2Vec	Pointer Network
Dai et al. [42]	TSP	DQN	Structure2Vec	-
Nazari et al. [9]	TSP, CVRP	REINFORCE with baseline	Elementwise projections	RNN with Attention
Xin et al. [47]	TSP, CVRP	REINFORCE with the greedy rollout	Pointer Network, Multi-head attention (Dynamic)	Pointer Network, Multi-head attention
Vera and Abad [59]	mCVRP	Actor-critic (A2C)	RNN	RNN with Attention
Zhang et al. [51]	mVRPSTW	REINFORCE with baseline	Multi-head attention	Multi-head attention
Kool et al. [44]	TSP, CVRP, OP, PCTSP, SDVRP, SPCTSP	REINFORCE with baseline	Multi-head attention	Multi-head attention
Peng et al. [46]	CVRP	REINFORCE with baseline	Graph attention network	Multi-head attention
Sykora et al. [62]	MAMP	REINFORCE	Linear projection	RNN with attention
Xin et al. [63]	TSP, CVRP, OP, PCTSP, SDVRP, SPCTSP	REINFORCE with the greedy rollout	Multi-head attention	Multi-head attention
Xu et al. [64]	TSP, CVRP, PCTSP, SDVRP	REINFORCE with baseline	Multi-head attention with gate aggregation	Self-attention with a attentive aggregation module
Kim et al. [53]	TSP, CVRP, PCTSP	REINFORCE with greedy rollout	Multi-head attention	Multi-head attention
Kwon et al. [54]	TSP, CVRP	REINFORCE with baseline	Multi-head attention	Multi-head attention
Emami and Ranka [55]	TSP	Deterministic Policy Gradient (DPG)	RNN	Sinkhorn layer
Joshi et al. [40]	TSP	REINFORCE with greedy rollout	Multi-head attention	Multi-head attention
Falkner and Schmidt-Thieme [60]	mCVRPTW	REINFORCE with greedy rollout	Self-attention, Linear projection	Multi-head attention
Joshi et al. [41]	TSP	REINFORCE with baseline	GNN	Multi-head attention
Li et al. [50]	CSP	REINFORCE with the greedy rollout	Pointer Network	RNN with attention
Drori et al. [65]	TSP, VRP	REINFORCE with baseline	Graph Attention Network (GAT)	Multi-head attention
Lin et al. [57]	EVRPTW	REINFORCE with greedy rollout	Structure2Vec	RNN with Attention
Li et al. [66]	MOTSP	Actor-critic (A2C)	1D-Conv	GRU with Attention
Li et al. [67]	PDP	REINFORCE with greedy rollout	Multi-head attention	Multi-head attention
Li et al. [68]	HCVRP	REINFORCE with greedy rollout	Multi-head attention	Feed-Forward networks, Multi-head attention
Duan et al. [69]	CVRP	REINFORCE with greedy rollout, SUPERVISE with policy-sampling	Graph Convolutional Networks	GRU with context-based attention, Multi-layer Perceptron

[74] presents a model to use reinforcement learning for neighborhood search to solve CVRP and split delivery VRP (SDVRP). In particular, a feasible solution is destroyed by a destroy operator in several locations. Then 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, 44] for VRP and SDVRP for different node sizes and show competitive results with LKH3, unified hybrid genetic search (UHGS). Also, [75] proposes using [9] model combined with Large Neighborhood Search (LNS) to solve VRPWT in ride-hailing services. They train the neural networks using supervised learning for the insertion operation inside LNS. [76] further enhances the idea of [77] 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 in order to maximize objective function by picking the actions presented as different operations available from Variable Neighborhood Search.

Along with approximated dynamic programming methods used to solve VRPs [78, 79], the combination of reinforcement learning with dynamic programming is an emerging research direction. For instance, [80] proposes to combine reinforcement learning with constraint programming (CP) using a dynamic programming approach to solve TSP, VRP, and TSPTW. Deep Q-learning and Proximal Policy Optimization (PPO) are integrated into CP search algorithms. [81] aims to mitigate the curse of dimensionality present in dynamic programming by restricting its search space to policies produced by learning methods. Identical to [37], the study generates the heatmap for the edges of a given graph that indicate the probabilities of entering those edges into the solution. Then this heatmap is used for branching in forwarding Dynamic Programming. [82, 83] uses neural networks to approximate the value functions for dynamic programming, which expedites the solution time. [84] 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. Then for policy evaluation, they used Mixed Integer Programming (MIP) to select the actions, combining the learned value functions. The resulting MIP is solved using the branch-and-cut approach of [85].

Another set of studies uses learning methods to pick heuristics to solve routing problems. For instance, [86, 87] 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 are passed to the decoder. The decoder produces the matrix of probabilities to select the pair of nodes to use for local operation. [88] proposes 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 reinforcement learning to solve TSP and CVRP. Similarly, [89] proposes to learn a general heuristic to solve VRP, VRPWT that given an initial solution destroys some part of the solution and learns to reconstruct an improved version of it. The study modify GNNs to embed both the nodes and edges, which passed through GRU to define the order of the nodes of the new repaired part for the solution. Also, [90] proposes to delegate subproblems of VRPs to black-box solvers and trains a subproblem generator model based on transformers and supervised learning. 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, [91] 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, [92] used [9]' model with an adaptive critic later combined with local search algorithms such as OR-tools and LNS. Also, [93] introduces Graph Pointer Nets (GPNs) to solve TSP, where a graph encoder made of GNNs is combined with Pointer Networks. Then 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 TSPWT and demonstrate comparable results with the construction heuristics. Another study by [94] 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 TSPWTR and update the parameters of NNs using the baseline rollout.

Learning methods can also be combined with prescriptive methods to solve complex routing problems. For instance, [95] 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. For instance, [96] proposes the

idea of training TSP in a supervised manner in small instances using LKH as the optimal solution and then using 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. Then they take a larger graph and split it into several graphs with size m, where each node can be present in several graphs. Then 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 Monte-Carlo Tree Search to improve further the solution, similar to [97]. The study solves TSPs with up to 10,000 nodes in a reasonable time.

V. Single vs. Multiple VRP Formulations

The majority of VRPs in practice involve routing either multiple heterogeneous or homogeneous vehicles [110]. 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. 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, $\mathbb{V} \setminus v_0$. We aim to find a sequence of nodes to be visited by a vehicle to minimize the total costs defined as the total distance travelled. We can further add capacity constraints by setting the maximum load, l_{max}, to be carried by a vehicle and let d_v represent a customer demand at node v. The problem naturally fits a single-agent reinforcement learning, where a vehicle represents the agent, and the environment is a simulated CVRP. Then the agent is trained to select action $A_{t+1} \in \mathbb{V}$ at each time step t, representing the node index to be visited next given the current state S_t. In CVRP with a single vehicle, the current state 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. This state representation allows to fully capture the dynamics of the problem and make the future decisions based only on the current state, thus preserving the Markov property. At each time step t, we can update the remaining load as follows:

$$l_{t+1} = l_{\text{max}} \quad \text{if } a_t = v_0 \quad \text{and} \quad l_{t+1} = l_t - d_{a_t} \quad \text{otherwise.}$$

Similarly, set Y_t includes all served customers with $d_v^t = 0$ for all $v \in \mathbb{V}$, which is updated based on the decision of the agent at time t:

$$d_v^{t+1} = d_v^t \quad \text{if } a_t \neq v \quad \text{and} \quad d_v^{t+1} = d_v^t - l_t \quad \text{otherwise.}$$

For each decision at time t, we calculate intermediate reward r_t defined as the distance between the current location, a_t and to be visited node a_{t+1} from which we can derive the total reward, R:

$$R = \sum_{t=0}^{T} r_t = \sum_{t=0}^{T-1} c_{a_t, a_{t+1}} \tag{25}$$

The majority of the studies focus on routing a single vehicle with some modification to the above CVRP by considering time window constraints [77, 92, 108], allowing split deliveries [9, 74], introducing dynamic demand [56] or generalizing to various forms of TSP [44, 52, 93, 94]. However, comparing the proposed models with other learning-based or non-learning methods is a challenging task due to the absence of widely-accepted benchmark instances and differences in the hardware settings. Also, most of the studies, except a few, using only randomly generated instances, which prevents to fairly compare them with non-learning methods that have been widely tested on real-world datasets.

Tables IV and V present the comparison of the surveyed studies to solve Euclidean distance TSP and CVRP, respectively, on randomly generated graphs with 20, 50, and 100 nodes. The results include the averages of objective values, gaps, and computational time for 10,000 instances if not indicated otherwise. The gaps are measured against the best-performing method.

We compare end-to-end learning methods [37, 42, 44, 53, 54, 63], hybrid methods [74, 81, 86, 87, 91, 101, 111] in their different configurations and used Concorde, LKH and OR-tools as baselines. We present the results reported in the original studies. The hybrid model of [86] performs slightly better both to solve TSP and CVRP than the rest of presented studies. However, it is extremely challenging to compare the computational times due to the differences in the hardware types and quantities and the number of batches used for inference. Also, [53] placed a time budget in the experiments, resulting in small running times. Tables VIII and IX show the generalization capacity of [44, 53, 54, 86, 87] on well-known TSPLib [112] and CVRPLib [113] datasets. We report the results of [44, 54] based on [86], the results of [111] are based on [53] and we report the results of [53, 86, 87] from the original studies. [93] performs the best across both datasets.

Solving large-scale routing problems using learning methods is still in development. One of such studies [90] presents the hybrid model that outperforms the hand-crafted heuristics such as Random, Count-based, and Max-min. Table VI reported from [90] shows the comparison with LKH, OR-tools and learning-based studies [44, 88]. Even though there have been some studies to solve VRPTW [51, 77, 92], the settings of the problem are different for each study, which prevents the comparisons between them.

TABLE III: The summary of studies with hybrid methods. CVRP-Capacitated VRP, CVRPTW-CVRP with Time Windows, MARL-Multi-agent Reinforcement Learning, 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

Name	Problems	Type	Learning Method	Non-learning method
Dorigo and Gambardella [71]	TSP	First	Q-Learning	Ant-colony
Liu and Zeng [72]	TSP	First	Q-learning	Genetic Algorithm
Alipour et al. [73]	TSP	First	MARL	Genetic Algorithm, 2-opt
Hottung and Tierney [74]	CVRP, SDVRP	First	Policy-based RL	Neighborhood Search
Syed et al. [75]	VRPTW in ride-hailing services	First	Supervised Learning	Large Neighborhood Search
Fernandes et al. [77]	VRPTW	First	Q-learning	Adaptive Local Search
Cappart et al. [80]	TSPTW	First	DQN, Proximal Policy Optimization (PPO)	Constraint Programming, Dynamic Programming
Kool et al. [81]	TSP, CVRP, TSPTW	First	Supervised learning	Dynamic Programming
Xu et al. [82]	TSP	First	Unsupervised learning	Dynamic Programming
Delarue et al. [84]	CVRP	First	Value-based RL	Mixed Integer Programming (MIP)
Ma et al. [86]	CVRP	First	The actor-critic variant of PPO	2-opt, swap and insert
Chen and Tian [88]	CVRP	First	Actor-Critic	Halide rewriter [98]
Li et al. [90]	Large-scale CVRP	First	Supervised Learning	MIP solver
Deudon et al. [91]	TSP	Second	MHA-based encoder, Transformers Decoder	2-opt local search
Zhao et al. [92]	CVRP and VRPTW in large scale	Second	Policy Gradient	OR-tools and LNS
Ma et al. [93]	TSPTW	Second	Hierarchical Policy based RL	Local search
Zhang et al. [94]	TSPTWR	Second	Policy based RL	Tabu search
Lin et al. [95]	Large-scale fleet management problem	Second	Contextual DQN and Contextual Actor-Critic	Linear programming
Chen et al. [99]	SDDPVD	First	Deep Q-learning	A policy function approximation
Tyasnurita et al. [100]	Open VRP	First	Hyper-heuristic classification	Modified Choice Function All Moves
Xing and Tu [97]	TSP	Second	Supervised Learning to learn heat maps	Monte Carlo Tree Search
Fu et al. [96]	TSP	Second	Supervised Learning to learn heat maps	Monte Carlo Tree Search
Hottung et al. [101]	TSP, CVRP	Second	Supervised Learning to learn latent space	Unconstrained continuous optimization
Gutierrez-Rodríguez et al. [102]	mVRPTW	Second	Supervised Learning	HMOEA-06 [103], MOGA-06 [104], MMOEAD-15 [105], HMPSO-16 [106]
da Costa et al. [48]	TSP	Second	DRL, Policy Gradient	2-opt local search
Fu et al. [107]	TSP	First	MCTS with reinforcement learning	2-opt local search
Wu et al. [87]	TSP, CVRP	First	Policy gradient	Pairwise local operators
Xin et al. [108]	TSP, CVRP, CVRPTW	First	Supervised Learning	LKH Algorithm
Yang et al. [83]	TSP	First	Deep Q-Learning	Dynamic Programming
Gao et al. [89]	CVRP, CVRPTW	Second	Actor-Critic	Very Large-scale Neighborhood Search
Joe and Lau [109]	DVRP	First	TD learning	Simulated Annealing
Gambardella and Dorigo [70]	TSP, ATSP	First	Q-learning	Ant System

TABLE IV: The performance comparison to solve TSP on random instances.

Method	TSP20			TSP50			TSP100		
	Obj.	Gap	Time	Obj.	Gap	Time	Obj.	Gap	Time
Concorde	3.83	-	3m	5.70	-	10m	7.76	-	1h
LKH	3.83	0.00%	38s	5.70	0.00%	5m	7.76	0.00%	20m
OR-Tools	3.86	0.94%	42s	5.85	2.87%	5m	8.06	3.86%	23m
Joshi et al. [37], GCN-BS	3.84[1]	0.01%	12m	5.70	0.01%	18m	7.87	1.39%	40m
Dai et al. [42]	3.89	1.42%	-	5.99	5.16%	-	8.31	7.03%	-
Kool et al. [44], AM(N=1,280)	3.83	0.06%	14m	5.72	0.48%	47m	7.94	2.32%	1.5h
Kool et al. [44], AM(N=5,000)	3.83	0.04%	47m	5.72	0.47%	2h	7.93	2.18%	5.5h
Kim et al. [53], AM+LCP {640,10}	3.84[1]	**0.00%**	0.18s	5.70	0.13%	0.30s	7.86	1.25%	0.57s
Kim et al. [53], AM+LCP {1280,10}	-	-	-	5.70	0.10%	0.45s	7.85	1.13%	0.90s
Kim et al. [53], AM+LCP*{1280,45}	-	-	-	5.70	0.02%	2.48s	7.81	0.54%	4.30s
Kwon et al. [54], POMO	3.83	0.04%	1s	5.70[1]	0.21%	2s	7.80	0.46%	11s
Kwon et al. [54], POMO×8 augment	3.83	**0.00%**	3s	5.69[1]	0.03%	16s	7.78	0.15%	1m
Xin et al. [63], MDAM-BS	3.84[1]	**0.00%**	3m	5.70	0.03%	14m	7.79	0.38%	44m
Kool et al. [81], DPDP (100k)	-	-	-	-	-	-	7.77[1]	0.00%	3h
Ma et al. [86], DACT (T=1k)	3.83	0.04%	24s	5.70	0.14%	1m	7.89	1.62%	4m
Ma et al. [86], DACT(T=5k)	3.83	**0.00%**	2m	5.70	0.02%	6m	7.81	0.61%	18m
Ma et al. [86], DACT(T=10k)	3.83	**0.00%**	5m	5.70	0.01%	13m	7.79	0.37%	40m
Ma et al. [86], DACT×4 augment	3.83	**0.00%**	10m	5.70	**0.00%**	1h	7.77	**0.09%**	2.5h
Wu et al. [87], (T=1,000)	3.83	0.03%	12m	5.74	0.83%	16m	8.01	3.24%	25m
Wu et al. [87], (T=3,000)	3.83	**0.00%**	39m	5.71	0.34%	45m	7.91	1.85%	1.5h
Wu et al. [87], (T=5,000)	3.83	**0.00%**	1h	5.70	0.20%	1.5h	7.87	1.42%	2h
Deudon et al. [91], EAN {M:1280}	3.84	0.11%	-	5.77	1.28%	-	8.75	12.70%	-
Deudon et al. [91], 2OPT {M:1280}	3.84	0.09%	-	5.75	1.00%	-	8.12	4.64%	-
Hottung et al. [101]	-	**0.00%** [2]	11m[2]	-	0.02%[2]	22m[2]	-	0.34%	55m[2]
da Costa et al. [111]	3.84[1]	**0.00%**	15m	5.70	0.12%	29m	7.83	0.87%	41m

[1] the obj. values obtained by Concorde or LKH may be slightly different from DACT since the 10,000 instances are randomly generated.
[2] the obj. values, gaps, or time are obtained based on 2,000 instances in their original papers

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	6.14	0.00%	1h	10.38	0.00%	4h	15.68	0.00%	8h
OR-Tools	6.46	5.68%	2m	11.27	8.61%	13m	17.12	9.54%	46m
Nazari et al. [9]{M: 10}	6.40	4.31%	-	11.15	7.46%	-	16.96	8.18%	-
Kool et al. [44], AM{M: 1280}	6.25	1.90%	0.05s	10.62	2.40%	0.14s	16.23	3.52%	0.34s
Kool et al. [44], AM{M: 2560}	6.25	1.80%	0.06s	10.61	2.24%	0.31s	16.17	3.14%	0.75s
Kool et al. [44], AM{M: 7500}	6.24	1.65%	0.09s	10.59	2.06%	0.36s	16.14	2.91%	1.42s
Kim et al. [53] {640,1}	6.17	1.15%	0.07s	10.56	1.74%	0.15s	16.05	2.58%	0.30s
Kim et al. [53] {1280,1}	6.16	0.92%	0.09s	10.54	1.54%	0.20s	16.03	2.43%	0.45s
Kim et al. [53] {2560,1}	6.15	0.84%	0.14s	10.52	1.38%	0.31s	16.00	2.24%	0.77s
Kim et al. [53] {6500,1}	-	-	-	-	-	-	15.98	2.11%	1.73s
Kwon et al. [54]	6.17[1]	0.82%	1s	10.49	1.14%	4s	15.83	0.98%	19s
Kwon et al. [54]×8 augment	6.14[1]	0.21%	5s	10.42	0.45%	26s	15.73	0.32%	2m
Hottung and Tierney [74]{I: 2000}	6.19	0.88%	1.00s	10.54	1.54%	1.63s	16.00	1.97%	2.18s
Kool et al. [81] (100k)	-	-	-	-	-	-	15.69[1]	0.31%	6h
Ma et al. [86] (T=1k)	6.15	0.28%	33s	10.61	2.13%	2m	16.17	3.18%	5m
Ma et al. [86] (T=5k)	6.13	0.00%	3m	10.48	1.01%	8m	15.92	1.55%	23m
Ma et al. [86] (T=10k)	6.13	-0.04%	6m	10.46	0.79%	16m	15.85	1.12%	45m
Ma et al. [86]×6 augment	6.13	**-0.08%**	35m	10.39	**0.14%**	1.5h	15.71	**0.19%**	4.5h
Wu et al. [87] (T=1,000)	6.16[1]	0.48%	23m	10.71	3.16%	48m	16.30	3.89%	1h
Wu et al. [87] (T=3,000)	6.14[1]	0.19%	1h	10.55	1.65%	2h	16.11	2.73%	3h
Wu et al. [87] (T=5,000)	6.12[1]	-0.03%	2h	10.45	0.70%	4h	16.03	2.21%	5h
Hottung et al. [101]	6.14[2]	-	21m	10.40[2]	-	41m[2]	15.75[2]	-	1.5h[2]

[1] the obj. values obtained by Concorde or LKH may be slightly different from DACT since the 10,000 instances are randomly generated.
[2] the obj. values, gaps, or time are obtained based on 2,000 instances in their original papers

TABLE VI: The performance comparison to solve large-scale CVRP on random instances.

Method	CVRP500		CVRP1000		CVRP2000	
	Cost	Time	Cost	Time	Cost	Time
LKH3 (95%)	62	4.4min	120.02	18min	234.89	52min
LKH3 (30k)	61.87	30min	119.88	77min	234.65	149min
OR-Tools	65.59	15min	126.52	15min	244.65	15min
Kool et al. [44], AM sampling	69.08	4.70s	151.01	17.40s	356.69	32.29s
Kool et al. [44], AM greedy	68.58	25ms	142.84	56ms	307.86	147ms
Chen and Tian [88]	73.6	58s	136.29	2.3min	257.61	8.1min
Random	61.99	71s	120.02	3.2min	234.88	6.4min
Count-based	61.99	59s	120.02	2.1min	234.88	5.3min
Max Min	61.99	59s	120.02	2.5min	234.89	5.2min
Li et al. [90] (Short)	61.99	38s	119.87	1.5min	234.89	3.4min
Li et al. [90] (Long)	**61.7**	76s	**119.55**	3.0min	**233.86**	6.8min

TABLE VII: The performance comparison to solve VRPTW on the Solomon benchmark problems.

	Model	VRPTW20				VRPTW50			
		Cost	K	Dist	Tinf	Cost	K	Dist	Tinf
TW1	Falkner and Schmidt-Thieme [60], OR-Tools-AU	2577.08	4.18	643.77	0.22s	4344.88	6.30	1110.68	1.76s
	Falkner and Schmidt-Thieme [60], 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. [44], AM+TW(greedy)	3766.88	5.29	1264.40	0.05s	7189.43	9.42	2882.53	0.12s
	Kool et al. [44], AM+TW(sampl.)	3041.24	5.74	1202.64	0.12s	7327.09	11.92	2921.39	0.38s
	Kool et al. [44], AM+TW(t10240)	2750.06	5.27	1163.58	0.95s	6878.84	11.28	2865.30	3.08s
	Falkner and Schmidt-Thieme [60], JAMPR(greedy)	1862.40	2.25	966.74	0.10s	3055.94	5.42	1733.24	0.24s
	Falkner and Schmidt-Thieme [60], JAMPR(sampl.)	**1716.60**	2.29	965.42	0.86s	**2691.55**	4.03	1811.06	3.07s
TW2	Falkner and Schmidt-Thieme [60], OR-Tools-AU	635.06	4.23	635.06	0.37s	1123.82	6.72	1123.82	3.81s
	Falkner and Schmidt-Thieme [60], 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. [44], AM+TW(greedy)	7615.69	2.00	1094.39	0.05s	40245.40	2.00	2687.95	0.12s
	Kool et al. [44], AM+TW(sampl.)	1572.31	6.56	1221.09	0.11s	7712.35	9.34	2953.65	0.34s
	Kool et al. [44], AM+TW(t10240)	1387.30	6.46	1170.49	0.90s	6730.55	9.99	2954.76	2.70s
	Falkner and Schmidt-Thieme [60], JAMPR(greedy)	674.72	4.32	626.80	0.11s	1273.20	6.02	1126.74	0.25s
	Falkner and Schmidt-Thieme [60], JAMPR(sampl.)	**620.68**	4.19	602.33	0.92s	**1116.76**	5.64	1076.79	2.32s
TW3	Falkner and Schmidt-Thieme [60], OR-Tools-AU	1317.81	4.07	637.15	1.07s	2707.72	6.12	1121.57	20.63s
	Falkner and Schmidt-Thieme [60], 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. [44], AM+TW(greedy)	3101.21	2.00	1094.39	0.05s	26467.34	2.00	2687.95	0.12s
	Kool et al. [44], AM+TW(sampl.)	1412.16	3.42	951.92	0.12s	4161.24	7.15	2674.03	0.34s
	Kool et al. [44], AM+TW(t10240)	1318.56	3.23	899.83	0.88s	3941.47	6.77	2575.28	2.67s
	Falkner and Schmidt-Thieme [60], JAMPR(greedy)	1002.81	1.00	733.01	0.10s	3158.26	2.01	1347.72	0.23s
	Falkner and Schmidt-Thieme [60], 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.

Therefore, in Table VII we report the comparison between [60] and [44] on the well-known Solomon dataset [114].

B. Multi-vehicle Routing

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

In *a centralized controller* setting, the controller is the agent, which takes 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 from 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. Then the central controller is trained to efficiently construct routes for all vehicles denoted as $p = [p_0, \ldots, p_T]$, where $p_t = [a^1_t, \ldots, a^M_t]$. Then 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:

$$R = \min_{p}\{\max\{c^1(p^1), \ldots, c^m(p^m)\}\} \tag{26}$$

TABLE VIII: The performance comparison on TSPLib.

Instance	Wu et al. [87] (T=3k)	Ma et al. [86] DACT (T=3k)	OR Tools	Kool et al. [44] (N=10k)	Kwon et al. [54] POMO ×8 augment	Wu et al. [87] (T=3k, M=1k)	Ma et al. [86] (T=10k)	Ma et al. [86] ×4 augment	da Costa et al. [111]	Kim et al. [53] 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%

TABLE IX: The performance comparison on CVRPLib.

Instance	Depot Type	Customer Type	Wu et al. [87] (T=5k)	Ma et al. [86] (T=5k)	OR Tools	Kool et al. [44] (N=10k)	Kwon et al. [54] ×8 augment	Wu et al. [87] (T=5k, M=100)	Ma et al. [86] (T=10k)	Ma et al. [86] ×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%**

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 travelling 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, which allows it to promote coordinated routing for the common goal of both homogeneous fleets and heterogeneous vehicles. For instance, all vehicles have common action space for multiple vehicles operating in a shared graph. Thus, without coordination, several vehicles may visit the same customer. The central controller, which assigns actions for each vehicle prevents such inefficiencies. For instance, [8, 25, 59, 115] and [57, 94, 102] use the centralized controller to solve mCVRP and mVRPWT with homogeneous set of vehicles. [56] and [95] propose deep reinforcement learning to route multiple vehicles with uncertain demand. Another emerging topic is to route a set of heterogeneous vehicles such as trucks and drones [88, 116] that also rely on the centralized controller.

On the other hand, MARL aims to train fully independent agents which cooperate together for a common goal. Formally each agent m has its local observation denoted as o^m_t, which only includes partial information about the environment. Then each agent is trained to select action A^m_{t+1} given its observation o^m_t and its state s^m_t. In order to promote coordinated routing, communication messages are allowed between agents. Then each agent is trained to construct its route $p^m = [a^m_0, \ldots, a^m_T]$, that contributes to the common reward:

$$R = \sum_{m=1}^{M} c^m(p^m) \tag{27}$$

where $c^m(p^m)$ is a cost function. The general framework to apply MARL for routing multiple vehicles is presented in [76]. The MARL formulations are suitable in dynamic environments when the customer demand is observed over time. For instance, in the case of online routing, vehicles trained in a MARL setting have the ability to make independent decisions to serve customers. For instance, [62] presents the MARL formulation to solve Multi-agent Mapping Problem to satisfy demand in known locations with unknown quantities. Table X summarizes studies focused on routing multiple vehicles.

VI. The Future Research Directions

This section discusses future research directions based on the presented survey.

a) Generalization: The generalization of learning methods to problems of different graph sizes is one of the primary and common challenges faced by many proposed models. For instance, to produce competitive solutions, separate sets of models need to be trained from scratch for each graph size [9, 42, 44, 45]. This leads to a substantial total training time, which can be sped up by advanced hardware equipped with multiple GPUs. Nevertheless, developing training techniques to transfer learning from small-sized graphs to large-sized graphs has not been studied extensively except in a few studies. [96] proposes to learn heatmaps for TSP on small sizes and apply them to arbitrarily large instances. Also, [41] investigates the generalization capabilities of supervised and reinforcement learning methods in end-to-end settings on TSP. The paper has shown that reinforcement learning is better than supervised learning because it does not learn from an existing solution. The study demonstrates that models trained on various graph sizes generalize better than the models trained on solo-sized graphs.

Scaling to large instances is an especially severe issue for end-to-end methods because training such models requires a large time budget and efficient hardware. On the contrary, hybrid models have demonstrated the potential to handle large-scale routing problems. For instance, [97] presents a model to solve TSP with 1,000 nodes. [90] speeds up the current solvers to solve VRPs with 500 and up to 3,000 nodes. [89]' experiments indicate the competitiveness of their hybrid method with the construction heuristics for VRPWT with 400 nodes.

Even though presenting the performance of the proposed models on real-world datasets has become more common in the recent years [8, 53, 87, 93, 116], the generalization of the learning-based models to real-world data is not well studied yet. Recently there have been some attempts [118] to establish well-accepted benchmark libraries and guidelines to compare the computational results of machine learning models to solve combinatorial optimization problems with the operations research methods. However, the fair comparison of the learning-based models among themselves and against non-learning-based methods still remains a complex and challenging task.

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 to solve routing problems. However, the solutions produced by learning methods in the best cases are comparable with the existing non-learning methods as reported in Tables IV, V, VI and VII. 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 and distances. However, practically important VRPs focus on the characteristics of the road networks such as costs of edges [119] 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. The comparison studies investigating properties of deep learning models for specific tasks such as graph embedding and a solution decoding for VRPs are needed.

TABLE X: The summary of studies to solve multi-vehichle 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. [56]	Online DVRP		✓	✓			✓
Vera and Abad [59]	mCVRP		✓	✓		✓	
Zhang et al. [51]	mVRP with soft TW		✓		✓	✓	
Sykora et al. [62]	Multi-agent Mapping Problem	✓			✓	✓	
Lin et al. [95]	online ride-sharing		✓		✓		✓
Falkner and Schmidt-Thieme [60]	mCVRPTW		✓		✓	✓	
Qin et al. [115]	VRP		✓	✓		✓	
Silva et al. [76]	mVRPTW	✓			✓	✓	
Van Knippenberg et al. [38]	mCVRP		✓		✓	✓	
Bogyrbayeva et al. [8]	mCVRP with charging		✓		✓	✓	
Bogyrbayeva et al. [116]	TSP with Drone		✓	✓		✓	
Chen et al. [99]	SDDPVD		✓	✓			✓
Lin et al. [57]	mEVRPTW		✓		✓	✓	
Gutierrez-Rodríguez et al. [102]	mVRPTW		✓		✓	✓	
Bono et al. [117]	mDSCVRPTW, mSCVRPTW		✓		✓		✓
Li et al. [68]	MMHCVRP, MSHCVRP		✓	✓			✓

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 [78]. For instance, online routing has become more critical with the increased demand for on-demand services such as parcel delivery [56, 99], ride-sharing [62, 95] and fractional ownership [120, 121]. 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.

However, routing multiple vehicles even with static inputs is challenging [8]. The majority of the surveyed studies focus on routing either a single vehicle or designed to solve the problems with the 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.

In summary, learning-based methods, either in their pure or hybrid forms, possess advantages such as solution speed and incorporating uncertainty in overcoming challenges faced by the urban transportation systems through solving vehicle routing problems.

REFERENCES

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

[2] Y. Yuan, D. Cattaruzza, M. Ogier, and F. Semet, "The last mile delivery problem," in *ROUTE 2018-International Workshop on Vehicle Routing, Intermodal Transportation and Related Areas*, 2018.

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

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

[5] G. Laporte, "The vehicle routing problem: An overview of exact and approximate algorithms," *European journal of operational research*, vol. 59, no. 3, pp. 345–358, 1992.

[6] K. F. Doerner and V. Schmid, "Survey: matheuristics for rich vehicle routing problems," in *International Workshop on Hybrid Metaheuristics*. Springer, 2010, pp. 206–221.

[7] M. Asghari, S. M. J. M. Al-e *et al.*, "Green vehicle routing problem: A state-of-the-art review," *International Journal of Production Economics*, vol. 231, p. 107899, 2021.

[8] A. Bogyrbayeva, S. Jang, A. Shah, Y. J. Jang, and C. Kwon, "A reinforcement learning approach for rebalancing electric vehicle sharing systems," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[9] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," *Advances in neural information processing systems*, vol. 31, 2018.

[10] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: a methodological tour d'horizon," *European Journal of Operational Research*, vol. 290, no. 2, pp. 405–421, 2021.

[11] U. Ritzinger, J. Puchinger, and R. F. Hartl, "A survey on dynamic and stochastic vehicle routing problems," *International Journal of Production Research*, vol. 54, no. 1, pp. 215–231, 2016.

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

[13] N. Mazyavkina, S. Sviridov, S. Ivanov, and

E. Burnaev, "Reinforcement learning for combinatorial optimization: A survey," *Computers & Operations Research*, vol. 134, p. 105400, 2021.

[14] 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. 120 388–120 416, 2020.

[15] Q. Wang, Y. Ma, K. Zhao, and Y. Tian, "A comprehensive survey of loss functions in machine learning," *Annals of Data Science*, vol. 9, no. 2, pp. 187–212, 2022.

[16] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3, pp. 279–292, 1992.

[17] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[18] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3, pp. 229–256, 1992.

[19] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[20] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.

[21] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[22] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[23] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[24] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[25] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[26] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[27] F. De Nijs, E. Walraven, M. de Weerdt, and M. Spaan, "Bounding the probability of resource constraint violations in multi-agent mdps," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[28] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[29] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[30] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2681–2690.

[31] P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 33, no. 6, pp. 750–797, 2019.

[32] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *Handbook of Reinforcement Learning and Control*, pp. 321–384, 2021.

[33] M. Held and R. M. Karp, "A dynamic programming approach to sequencing problems," *Journal of the Society for Industrial and Applied mathematics*, vol. 10, no. 1, pp. 196–210, 1962.

[34] M. Hahsler and K. Hornik, "Tsp-infrastructure for the traveling salesperson problem," *Journal of Statistical Software*, vol. 23, no. 2, pp. 1–21, 2007.

[35] K. Helsgaun, "An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems," *Roskilde: Roskilde University*, pp. 24–50, 2017.

[36] 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 *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 4731–4738.

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

[38] M. Van Knippenberg, M. Holenderski, and V. Menkovski, "Complex vehicle routing with memory augmented neural networks," in *2020 IEEE Conference on Industrial Cyberphysical Systems (ICPS)*, vol. 1. IEEE, 2020, pp. 303–308.

[39] A. Milan, S. H. Rezatofighi, R. Garg, A. Dick, and I. Reid, "Data-driven approximations to np-hard problems," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

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

[41] C. K. Joshi, Q. Cappart, L.-M. Rousseau, and T. Laurent, "Learning tsp requires rethinking generalization," in *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für

Informatik, 2021.

[42] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[43] M. Riedmiller, "Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method," in *European conference on machine learning*. Springer, 2005, pp. 317–328.

[44] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!" 2019.

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

[46] B. Peng, J. Wang, and Z. Zhang, "A deep reinforcement learning algorithm using dynamic attention model for vehicle routing problems," in *International Symposium on Intelligence Computation and Applications*. Springer, 2019, pp. 636–650.

[47] L. Xin, W. Song, Z. Cao, and J. Zhang, "Step-wise deep learning models for solving routing problems," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4861–4871, 2020.

[48] P. R. d. O. da Costa, J. Rhuggenaath, Y. Zhang, and A. Akcay, "Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning," in *Asian Conference on Machine Learning*. PMLR, 2020, pp. 465–480.

[49] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[50] K. Li, T. Zhang, R. Wang, Y. Wang, Y. Han, and L. Wang, "Deep reinforcement learning for combinatorial optimization: Covering salesman problems," *IEEE Transactions on Cybernetics*, 2021.

[51] 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," *Transportation Research Part C: Emerging Technologies*, vol. 121, p. 102861, 2020.

[52] L. Xin, W. Song, Z. Cao, and J. Zhang, "Multi-decoder attention model with embedding glimpse for solving vehicle routing problems," in *Proceedings of 35th AAAI Conference on Artificial Intelligence*, 2021, pp. 12 042–12 049.

[53] M. Kim, J. Park *et al.*, "Learning collaborative policies to solve np-hard routing problems," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[54] Y.-D. Kwon, J. Choo, B. Kim, I. Yoon, Y. Gwon, and S. Min, "Pomo: Policy optimization with multiple optima for reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 188–21 198, 2020.

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

[56] J. James, W. Yu, and J. Gu, "Online vehicle routing with neural combinatorial optimization and deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 10, pp. 3806–3817, 2019.

[57] B. Lin, B. Ghaddar, and J. Nathwani, "Deep reinforcement learning for the electric vehicle routing problem with time windows," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[58] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[59] J. M. Vera and A. G. Abad, "Deep reinforcement learning for routing a heterogeneous fleet of vehicles," in *2019 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*. IEEE, 2019, pp. 1–6.

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

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

[62] Q. Sykora, M. Ren, and R. Urtasun, "Multi-agent routing value iteration network," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9300–9310.

[63] L. Xin, W. Song, Z. Cao, and J. Zhang, "Multi-decoder attention model with embedding gli2mpse for solving vehicle routing problems," in *Proceedings of 35th AAAI Conference on Artificial Intelligence*, 2021.

[64] 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 Transactions on Cybernetics*, 2021.

[65] I. Drori, A. Kharkar, W. R. Sickinger, B. Kates, Q. Ma, S. Ge, E. Dolev, B. Dietrich, D. P. Williamson, and M. Udell, "Learning to solve combinatorial optimization problems on real-world graphs in linear time," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2020, pp. 19–24.

[66] K. Li, T. Zhang, and R. Wang, "Deep reinforcement learning for multiobjective optimization," *IEEE transactions on cybernetics*, vol. 51, no. 6, pp. 3103–3114, 2020.

[67] 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 Transactions on Intelligent Transportation Systems*, 2021.

[68] J. Li, Y. Ma, R. Gao, Z. Cao, A. Lim, W. Song, and J. Zhang, "Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem,"

IEEE Transactions on Cybernetics, pp. 1–14, 2021.

[69] L. Duan, Y. Zhan, H. Hu, Y. Gong, J. Wei, X. Zhang, and Y. Xu, "Efficiently solving the practical vehicle routing problem: A novel joint learning approach," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 3054–3063.

[70] L. M. Gambardella and M. Dorigo, "Ant-q: A reinforcement learning approach to the traveling salesman problem," in *Machine learning proceedings 1995*. Elsevier, 1995, pp. 252–260.

[71] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Transactions on evolutionary computation*, vol. 1, no. 1, pp. 53–66, 1997.

[72] F. Liu and G. Zeng, "Study of genetic algorithm with reinforcement learning to solve the tsp," *Expert Systems with Applications*, vol. 36, no. 3, pp. 6995–7001, 2009.

[73] 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 Computing and Applications*, vol. 30, no. 9, pp. 2935–2951, 2018.

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

[75] A. A. Syed, K. Akhnoukh, B. Kaltenhaeuser, and K. Bogenberger, "Neural network based large neighborhood search algorithm for ride hailing services," in *EPIA Conference on Artificial Intelligence*. Springer, 2019, pp. 584–595.

[76] 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," *Expert Systems with Applications*, vol. 131, pp. 148–171, 2019.

[77] 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 *2009 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 2009, pp. 3071–3076.

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

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

[80] Q. Cappart, T. Moisan, L.-M. Rousseau, I. Prémont-Schwarz, and A. A. Cire, "Combining reinforcement learning and constraint programming for combinatorial optimization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021, pp. 3677–3687.

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

[82] S. Xu, S. S. Panwar, M. Kodialam, and T. Lakshman, "Deep neural network approximated dynamic programming for combinatorial optimization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 02, 2020, pp. 1684–1691.

[83] F. Yang, T. Jin, T.-Y. Liu, X. Sun, and J. Zhang, "Boosting dynamic programming with neural networks for solving np-hard problems," in *Asian Conference on Machine Learning*. PMLR, 2018, pp. 726–739.

[84] A. Delarue, R. Anderson, and C. Tjandraatmadja, "Reinforcement learning with combinatorial actions: An application to vehicle routing," *Advances in Neural Information Processing Systems*, vol. 33, pp. 609–620, 2020.

[85] R. Anderson, J. Huchette, W. Ma, C. Tjandraatmadja, and J. P. Vielma, "Strong mixed-integer programming formulations for trained neural networks," *Mathematical Programming*, vol. 183, no. 1, pp. 3–39, 2020.

[86] Y. Ma, J. Li, Z. Cao, W. Song, L. Zhang, Z. Chen, and J. Tang, "Learning to iteratively solve routing problems with dual-aspect collaborative transformer," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[87] Y. Wu, W. Song, Z. Cao, J. Zhang, and A. Lim, "Learning improvement heuristics for solving routing problems," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[88] X. Chen and Y. Tian, "Learning to perform local rewriting for combinatorial optimization," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

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

[90] S. Li, Z. Yan, and C. Wu, "Learning to delegate for large-scale vehicle routing," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[91] M. Deudon, P. Cournut, A. Lacoste, Y. Adulyasak, and L.-M. Rousseau, "Learning heuristics for the tsp by policy gradient," in *International conference on the integration of constraint programming, artificial intelligence, and operations research*. Springer, 2018, pp. 170–181.

[92] J. Zhao, M. Mao, X. Zhao, and J. Zou, "A hybrid of deep reinforcement learning and local search for the vehicle routing problems," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

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

[94] R. Zhang, A. Prokhorchuk, and J. Dauwels, "Deep reinforcement learning for traveling salesman problem with time windows and rejections," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[95] K. Lin, R. Zhao, Z. Xu, and J. Zhou, "Efficient large-scale fleet management via multi-agent deep reinforcement learning," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1774–1783.

[96] Z.-H. Fu, K.-B. Qiu, and H. Zha, "Generalize a small pre-trained model to arbitrarily large tsp instances," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 8, 2021, pp. 7474–7482.

[97] Z. Xing and S. Tu, "A graph neural network assisted monte carlo tree search approach to traveling salesman problem," *IEEE Access*, vol. 8, pp. 108 418–108 428, 2020.

[98] 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.

[99] X. Chen, M. W. Ulmer, and B. W. Thomas, "Deep q-learning for same-day delivery with vehicles and drones," *European Journal of Operational Research*, vol. 298, no. 3, pp. 939–952, 2022.

[100] R. Tyasnurita, E. Özcan, and R. John, "Learning heuristic selection using a time delay neural network for open vehicle routing," in *2017 IEEE Congress on Evolutionary Computation (CEC)*. Ieee, 2017, pp. 1474–1481.

[101] A. Hottung, B. Bhandari, and K. Tierney, "Learning a latent search space for routing problems using variational autoencoders," in *International Conference on Learning Representations*, 2020.

[102] 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," *Expert Systems with Applications*, vol. 118, pp. 470–481, 2019.

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

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

[105] 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," *Computers & Operations Research*, vol. 62, pp. 61–77, 2015.

[106] D. Wu, M. Dong, H. Li, and F. Li, "Vehicle routing problem with time windows using multi-objective co-evolutionary approach," *International Journal of Simulation Modelling*, vol. 15, no. 4, pp. 742–753, 2016.

[107] Z.-H. Fu, K.-B. Qiu, M. Qiu, and H. Zha, "Targeted sampling of enlarged neighborhood via monte carlo tree search for tsp," 2019.

[108] 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," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[109] W. Joe and H. C. Lau, "Deep reinforcement learning approach to solve dynamic vehicle routing problem with stochastic customers," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 30, 2020, pp. 394–402.

[110] Z. Sun, M. Greiff, A. Robertsson, and R. Johansson, "Feasible coordination of multiple homogeneous or heterogeneous mobile vehicles with various constraints," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 1008–1013.

[111] P. da Costa, J. Rhuggenaath, Y. Zhang, A. Akcay, and U. Kaymak, "Learning 2-opt heuristics for routing problems via deep reinforcement learning," *SN Computer Science*, vol. 2, no. 5, pp. 1–16, 2021.

[112] G. Reinelt, "Tsplib—a traveling salesman problem library," *ORSA journal on computing*, vol. 3, no. 4, pp. 376–384, 1991.

[113] E. Uchoa, D. Pecin, A. Pessoa, M. Poggi, T. Vidal, and A. Subramanian, "New benchmark instances for the capacitated vehicle routing problem," *European Journal of Operational Research*, vol. 257, no. 3, pp. 845–858, 2017.

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

[115] W. Qin, Z. Zhuang, Z. Huang, and H. Huang, "A novel reinforcement learning-based hyper-heuristic for heterogeneous vehicle routing problem," *Computers & Industrial Engineering*, vol. 156, p. 107252, 2021.

[116] 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," *arXiv preprint arXiv:2112.12545*, 2021.

[117] G. Bono, J. S. Dibangoye, O. Simonin, L. Matignon, and F. Pereyron, "Solving multi-agent routing problems using deep attention mechanisms," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 12, pp. 7804–7813, 2020.

[118] L. Accorsi, A. Lodi, and D. Vigo, "Guidelines for the

computational testing of machine learning approaches to vehicle routing problems," *Operations Research Letters*, vol. 50, no. 2, pp. 229–234, 2022.

[119] 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, 2021.

[120] A. Bogyrbayeva, M. Takalloo, H. Charkhgard, and C. Kwon, "An iterative combinatorial auction design for fractional ownership of autonomous vehicles," *International Transactions in Operational Research*, vol. 28, no. 4, pp. 1681–1705, 2021.

[121] M. Takalloo, A. Bogyrbayeva, H. Charkhgard, and C. Kwon, "Solving the winner determination problem in combinatorial auctions for fractional ownership of autonomous vehicles," *International Transactions in Operational Research*, vol. 28, no. 4, pp. 1658–1680, 2021.