MODRL/D-EL: Multiobjective Deep Reinforcement Learning with Evolutionary Learning for Multiobjective Optimization

1st Yongxin Zhang
School of Computer Science and Engineering
Sun Yat-sen University
Guangzhou, P. R. China
zhangyx266@mail2.sysu.edu.cn

2nd Jiahai Wang
School of Computer Science and Engineering
Sun Yat-sen University
Guangzhou, P. R. China
wangjiah@mail.sysu.edu.cn

3rd Zizhen Zhang
School of Computer Science and Engineering
Sun Yat-sen University
Guangzhou, P. R. China
zhangzzh7@mail.sysu.edu.cn

4th Yalan Zhou
College of Information
Guangdong University of Finance and Economics
Guangzhou, P. R. China
zhouylan@163.com

Abstract—Learning-based heuristics for solving combinatorial optimization problems has recently attracted much academic attention. While most of the existing works only consider the single objective problem with simple constraints, many real-world problems have the multiobjective perspective and contain a rich set of constraints. This paper proposes a multiobjective deep reinforcement learning with evolutionary learning algorithm for a typical complex problem called the multiobjective vehicle routing problem with time windows (MO-VRPTW). In the proposed algorithm, the decomposition strategy is applied to generate subproblems for a set of attention models. The comprehensive context information is introduced to further enhance the attention models. The evolutionary learning is also employed to fine-tune the parameters of the models. The experimental results on MO-VRPTW instances demonstrate the superiority of the proposed algorithm over other learning-based and iterative-based approaches.

I. INTRODUCTION

A multiobjective optimization problem (MOP) can be generally defined as follows,

$$\begin{align*}
\text{Min} & \quad F(x) = (f_1(x), f_2(x), \ldots, f_m(x))^T, \\
\text{s.t.} & \quad x \in X,
\end{align*}$$

(1)

where $X$ is the solution space, $F(x)$ is a $m$-dimensional vector in which $f_i$ is the $i$-th objective function of the MOP. Since these $m$ objective functions potentially conflict with each other, one would be keen in finding a set of trade-off solutions among which the most desirable solution can be chosen.

Formally, let $x,y \in \mathbb{R}^m$ be two objective vectors of MOP. $x$ is said to dominant to $y$, if $x_i \leq y_i$, $\forall i \in \{1,2,\ldots,m\}$ and $x_i < y_i$, $\exists i \in \{1,2,\ldots,m\}$, denoted as $x \prec y$. A solution $x^*$ is called Pareto optimal solution, if $F(x) \neq F(x^*)$, $\forall x \in X$. The set which contains all the Pareto optimal solutions is called the Pareto set ($PS$), and the set \{\{F(x)|x \in PS\}\} is called the Pareto frontier ($PF$).

In literature, many iterative-based algorithms are proposed to solve MOPs. These algorithms can be roughly divided into two categories: the multiobjective evolutionary algorithms (MOEAs) (e.g., MOEA/D [1] and NSGA-II [3]), and multiobjective local search based algorithms (e.g., MOGLS [4]). MOEAs search for PF by evolving a population of solutions simultaneously. MOGLS uses local search operators to improve solutions iteratively. These algorithms perform well in many multiobjective benchmark problems. However, as presented in [5], the iterative-based algorithms have some drawbacks. First, to find near-optimal solutions, a large number of iterations are required, so it would be very time consuming. Second, the iteration process needs to be re-performed once a slight change of problem happens or a similar instance is encountered. Third, most heuristic operators used in these algorithms heavily rely on experts’ empirical knowledge, which may affect the solution quality.

Recently, learning-based heuristics for solving optimization problems has attracted more and more attention [6–8]. In particular, deep reinforcement learning (DRL) has made remarkable achievement in solving single objective problems [9–13]. In addition, based on the decomposition strategy [1] and DRL algorithm, a multiobjective deep reinforcement learning (MODRL) framework [5] is proposed for MOPs. Following this framework, [14] proposes a multiobjective deep reinforcement learning algorithm with attention model.
(MODRL/D-AM), which achieves good performance on multiobjective traveling salesman problem. The characteristics of these works are summarized in Table I.

In this paper, we initiate the study on tackling an MOP with complex constraints, called multiobjective vehicle routing problem with time windows (MO-VRPTW). MO-VRPTW is more applicable in real-world scenarios. We try to apply learning-based heuristics for this problem. We have found that the attention model [13] in MODRL/D-AM constructs solution by sequentially adding nodes to partial solution. The selection of next node is based on very limited context information, which may lead to inefficiencies in MOPs (e.g., MO-VRPTW). Besides, the diversity of solutions obtained by MODRL/D-AM is not as good as expected [5]. To address these issues, this paper proposes a multiobjective deep reinforcement learning and evolutionary learning (MODRL/D-EL). In the proposed algorithm, two mechanisms are employed. First, the comprehensive context information (CCI) of MOPs is taken into consideration. The attention model with CCI can make more reasonable decision in constructing solution. Second, a hybrid training method composed of deep reinforcement learning and evolutionary learning (EL) is employed to train the models. The comparisons between our method and the related works are shown in Table I.

To summarize, the contributions of this paper are threefold:

- The attention model with comprehensive context information is introduced to solve MO-VRPTW. It is shown to be effective in improving the solution quality.
- A two-stage hybrid learning strategy is presented. In the first stage, DRL is employed to train models. In the second stage, evolutionary learning is used to fine-tune the parameters of models. The results indicate that both convergence and diversity of solutions can be improved.
- Computational experiments are conducted on both random instances and classical benchmark instances. It is demonstrated that the proposed method outperforms several existing methods.

The remaining of this paper is structured as follows. In Section II, problem formulation of MO-VRPTW is introduced. MODRL/D-AM is briefly reviewed in Section III. Thereafter, Section IV provides a detailed description of the proposed MODRL/D-EL. Experimental results are shown and analyzed in Section IV. Conclusions are drawn in Section V.

II. PROBLEM FORMULATION

MO-VRPTW is a classical MOP with complex constraints and has a high relevance for many practical applications. It can be defined on a complete undirected graph \( G = (V, E) \). Node set \( V = \{v_i | i = 0, 1, ..., N\} \), where node \( i = 0 \) is the depot and \( i \in \{1, ..., N\} \) are the customers. Each node \( v_i \) has some attributes including Euclidean coordinates \( v_i \in \mathbb{R}^2 \), a time window constraint \( t_i \in \mathbb{R}^2 \), and a demand \( q_i \in \mathbb{R}^+ \) that requires to be satisfied. The time window constraint \( t_i \) indicates a tuple \( [a_i, b_i] \), \( a_i \leq b_i \), where \( a_i \) and \( b_i \) are lower and upper bounds of the possible service time. Moreover, the service at each customer \( v_i \in V \) requires an identical service duration \( w \). For the depot, \( q_0 = 0 \) and \( t_0 = [a_0, b_0] \), where \( a_0 \) and \( b_0 \) denote the earliest possible departure from and latest return to the depot, respectively. \( E = \{e_{ij} | i, j \in V\} \) is the edge set of graph \( G \). The traveling distance of \( e_{ij} \) is defined as the Euclidean distance between \( v_i \) and \( v_j \), and the traveling time of \( e_{ij} \) is equal to the traveling distance numerically. There is a fleet contains \( K \) uniform vehicles with identical capacity \( Q > 0 \). Vehicle \( k \) departs from the depot, serves customers with respect to capacity and time windows constraints, and finally returns to the depot. The sequence of customers that the vehicle severed constitutes a route \( r_k \). A solution is a set of routes, denoted as \( s = \{r_1, r_2, ..., r_K\} \). A solution \( s \) needs to meet the following constraints: (1) all customers are visited exactly once, i.e., \( \bigcup_{k \in s} r_k = V \) and \( r_a \cap r_b = \emptyset, \forall a, b \in \{1, 2, ..., K\} \); (2) all routes satisfy the capacity constraint, i.e., \( \sum_{i \in r_k} q_i \leq Q \); (3) all customers are served within their specific time windows.

This paper focuses on two conflicting objectives, which are commonly adopted in previous work [15]: the total traveling distance and makespan, formally described as follows.

\[
\begin{align*}
\min \quad & F(s) = (f_1(s), f_2(s))^T, \\
& f_1(s) = \sum_{k=1}^{K} c_k, \\
& f_2(s) = \max(c_1, c_2, ..., c_K), \\
\text{s.t.} \quad & s \in S,
\end{align*}
\]

where \( c_k \) is the traveling distance of \( k \)-th route in solution \( s \). \( S \) is the valid solution space under the capacity and time windows constraints.

III. BRIEF REVIEW OF MODRL/D-AM

A. Decomposition Strategy

A well-known decomposition strategy [1] is employed in MODRL/D-AM for MOPs. Specifically, given a set of uniformly spread weight vectors \( \Lambda = \{\lambda_1, \lambda_2, ..., \lambda_M\} \), \( \lambda_i = (\lambda_{i1}, ..., \lambda_{iM})^T \), MODRL/D-AM adopts the weighted sum approach to convert MOP into \( M \) single objective subproblems. Then, a set of neural network models \( \pi = \{\pi_1, \pi_2, ..., \pi_M\} \) is trained for tackling these subproblems, where \( \pi_i \) corresponds to the subproblem with weight vector \( \lambda_i \).

The MO-VRPTW is decomposed into \( M \) single objective subproblems through decomposition strategy. For each subproblem, reinforcement learning is employed to train a neural network model to solve it. The states, actions, transition, rewards and policy in the reinforcement learning framework are defined as follows:

- **States**: a state consists of two parts. The first part includes the partial solution of subproblem and the candidate nodes. The second part is the position, remaining capacity, travel distance and travel time of each vehicle;
- **Actions**: an action is the next node to visit;
- **Transition**: transition corresponds to adding the selected action (node) into partial solution, and updating the state of corresponding vehicle;
In MODRL/D-AM, an attention model \cite{13} is employed for a subproblem. The attention model, say $\pi_i$, that selects a candidate node as next visiting node according to current state. The details of neural network model will be introduced in the following subsection.

### B. Model of subproblem: Attention Model

In MODRL/D-AM, an attention model \cite{13} is employed for a subproblem. The attention model, say $\pi_i$, is based on the encoder-decoder architecture, parametrized by $\theta$.

1) **Encoder:** The encoder first takes the node-wise features $x_i = (c_i, t_i, q_i), i \in V$, as input. It computes the initial embedding $h_i^{(0)} \in \mathbb{R}^{d_{emb}}$ ($d_{emb}$ is set to 128 empirically) of $v_i$ through a linear projection, i.e., $h_i^{(0)} = Wx_i + b$. Then the initial embedding $h_i^{(0)}$ is updated through a stack of three self-attention (SA) blocks, i.e.,

$$h_i = SA(SA(SA(h_i^{(0)}, H(0)))) \quad i \in V,$$

where $H(0) = (h_0^{(0)}, h_1^{(0)}, ..., h_N^{(0)})$ is the sequence of initial embeddings. SA block consists of two sublayers: a multi-head attention (MHA) and a node-wise fully connected feed-forward layer (FF). Each sublayer also includes a skip-head attention layer (MHA) and a node-wise fully connected embedding.

$$\hat{h}_i = BN^l(h_i^{(l-1)} + \text{MHA}^l(h_i^{(l-1)}, H^{(l-1)})),$$

$$h_i^{(l)} = BN^l(\hat{h}_i + \text{FF}^l(\hat{h}_i)), $$

where $h_i^{(l)}$ denotes the node embedding produced by block $l$. The multi-head attention layer is a linear combination of multiple (e.g., eight) single-head attention (SHA) layers, each of which takes a slice of all the input elements, formally defined as follows,

$$\text{MHA}(h, H; W) = \sum_{i=1}^{Z} W_i^{\text{head}} \text{SHA}(h_{\text{slice}(i)}, H;\text{slice}(i); W),$$

$$\text{SHA}(h, H; W) = \sum_{j=1}^{H} \text{attn}(h, H; W_{\text{query}}, W_{\text{key}}) W_{\text{value}} H_j,$$

where “\text{attn}” denotes the concatenation operator.

The decoder consists of a MHA layer and a subsequent layer with just attention weights operating on a masked input sequence. A valid solution under capacity and time windows constraints is promised by mask function. The mask function rules out the nodes that would violate constraints and the nodes that are already served by setting corresponding attention weight to $-\infty$, and thus yield probability of zero, i.e.,

$$p_\theta(v^t = i|s_{t-1}) = \pi_\theta(v^t = i|H, C^{t-1})$$

$$= \text{attn}(\text{MHA}(C^{t-1}, H), \text{mask}(H)),$$

where $s_{t-1}$ denotes the partial solution in time step $t - 1$.

The probability of a complete solution $s$ for instance $G$ can be modeled by the chain rule as follows,

$$p_\theta(s|G) = \prod_{t=1}^{T} \pi_\theta(v^t|H, C^{t-1}).$$

### C. Training Algorithm: Reinforcement Learning

Consider a specific subproblem with its model $\pi_\theta$, a well-known policy gradient based REINFORCE algorithm \cite{19} is employed to train it.
The gradients of parameters $\theta$ is formulated as follows,

$$\nabla_\theta J(\theta|G) = \mathbb{E}_{s \sim p(s|G)}[(\lambda^T F(s|G) - b(G))\nabla_\theta \log p_\theta(s|G)],$$

(13)

where $G$ is an input instance, $s$ is a complete solution, $\lambda$ is weight vector and $b(G)$ is a value function to reduce gradient variance. In the training process, the gradients of parameters $\theta$ can be approximated by Monte Carlo sampling as follows,

$$\nabla_\theta J(\theta|G) \approx \frac{1}{B} \sum_{j=1}^{B} [(\lambda^T F(s_j|G_j) - b(G_j))\nabla_\theta \log p_\theta(s_j|G_j)],$$

(14)

where $B$ is the batch size. For the training of $M$ subproblem models, the neighborhood-based parameter-transfer strategy [5] is employed to accelerate the training process.

IV. THE PROPOSED ALGORITHM: MODRL/D-EL

A. Motivation

The MODRL/D-AM performs well in simple MOPs (e.g., MO-TSP). In more complicated problems such as MO-VRPTW, the context embedding $C$ in the attention model only contains the information about the vehicle status and the last visited node in current route, while the nodes and vehicles in other routes are not taken into consideration. The limited context information usually leads to sub-optimal decisions. Besides, the diversity of solutions obtained by MODRL/D-AM is not as good as expected [5]. Therefore, this paper proposes to consider more comprehensive context information in constructing solution, and uses evolutionary learning to fine-tune the parameters of models to improve the diversity of solutions.

B. Model of Subproblem: Attention Model with Comprehensive Context Information

The proposed model of subproblem is also based on the attention mechanism [13]. The node encoder is consistent with the encoder in MODRL/D-AM. To provide comprehensive context information for the decoder to make more reasonable decision in constructing solution, vehicle encoder and route encoder are introduced, and the context embedding is extended as in [20]. The overall architecture of our model is shown in Fig. 1.

Both of the vehicle encoder and route encoder are multi-layer perceptron (MLP) and denoted as $g_v$ and $g_r$, respectively. $g_v$ has 3 layers and $g_r$ has 2 layers. $g_v$ encodes the vehicle feature $d_k$ which consists of vehicle index $k$, the current return distance to depot, the current position (given by coordinates of the last served node) and current time of vehicle $k$. $g_r$ encodes the route feature (given by the nodes served so far). Finally, the embedding of vehicle $k$, denoted as $h_k^{\text{vehicle}}$, is computed by aggregating the information of $g_v$ and $g_r$:

$$h_k^{\text{vehicle}} = [g_v(d_k); \frac{1}{|r_k|} \sum_{r_k} g_r(h_{i}^{\text{node}})].$$

(15)

The new context embedding also contains the information about the entire fleet and the last served nodes of vehicles, defined as follows,

$$h^{\text{fleet}} = \frac{1}{K} \sum_{k=1}^{K} h_k^{\text{vehicle}}, \quad h^{\text{last}} = \frac{1}{K} \sum_{k=1}^{K} h_{\text{last}(r_k)},$$

(16)

where $\text{last}(r_k)$ denotes the last node in route $r_k$. Finally, the context embedding is defined as follows,

$$C = [h^{\text{graph}}; h^{\text{fleet}}; h^{\text{cur}}; h^{\text{depot}}; h^{\text{last}}],$$

(17)

where $h^{\text{cur}}$ denotes the embedding of vehicle in current constructing route, and $h^{\text{depot}}$ denotes the embedding of depot.

To extract features of nodes and vehicles more effectively, the node embedding $h_i^{\text{node}}$ and vehicle embedding $h_i^{\text{vehicle}}$ are combined to create a new sequence of node embedding $H$, called joint combinatorial space [20] of all nodes and current vehicle, formally defined as follows,

$$H = \{g_a(h^{\text{cur}}, h_i^{\text{node}}) | i \in V\},$$

(18)

where $g_a$ is a neural network that computes compatibility between a vehicle and a node, formally defined as follows,

$$g_a(h^{\text{cur}}, h_i^{\text{node}}; W_1, W_2, W_3) = W_1 h_i^{\text{node}} + W_2 h^{\text{cur}} + W_3 (h^{\text{cur}} \odot h_i^{\text{node}}) (h_i^{\text{node}})^T,$$

(19)

where $\odot$ is the element-wise product.

Algorithm 1 Evolutionary Learning

Input: The initial population $\Pi = \{\pi_1, \pi_2, ..., \pi_M\}$ where $\pi_i$ has parameters $\theta_i$, training batch size $B$, maximum generation $D$

Output: optimized model set $\Pi^* = \{\pi_1^*, \pi_2^*, ..., \pi_M^*\}$

1: for $i = 1, ..., M$ do
2: \hspace{1cm} fitness$_{\pi_i} = \text{Evaluate}(\pi_i)$
3: end for
4: for $g = 1, ..., D$ do
5: \hspace{1cm} for $i = 1, ..., M$ do
6: \hspace{2cm} $G_j = \text{RandomInstance()} \forall j \in \{1, ..., B\}$ \hspace{1cm} \triangleright randomly sample training data
7: \hspace{2cm} $\epsilon = \sqrt{\sum_{k=0}^{N} (\sum_{j=1}^{B} \nabla \theta p(u^j = k|s_{t-1}, G_j))^2}$
8: \hspace{2cm} $x \sim N(0, \mu I)$
9: \hspace{2cm} $\theta_i = \theta_i + \frac{\epsilon}{x}$
10: \hspace{2cm} generate offspring $\hat{\pi}_i$ with parameters $\hat{\theta}_i$ and fitness$_{\pi_i} = \text{Evaluate}(\hat{\pi}_i)$
11: \hspace{2cm} end for
12: sort individuals (policies/models) with non-domination level and crowding distance, and select $M$ individuals for next generation
13: end for
C. Hybrid Training Algorithm with Evolutionary Learning

The hybrid training algorithm consists of two stages. In the first stage, the models of subproblems are trained by the well-known REINFORCE algorithm [19] as described in Section III-C. The reward of the subproblem is the weighted sum of two objectives as defined in (2) with weight vector $\lambda_i$.

In the second stage, the models trained by the first stage are taken as the initial population. Then they are evolved by variation operators. At each generation, each individual model generates an offspring by the proximal mutation [21], which takes SM-G-SUM [22] mutation operator to add scaled Gaussian perturbation to the model parameters $\theta$, i.e.,

$$\theta \leftarrow \theta + \frac{x}{\epsilon}, \quad x \sim N(0, \mu I),$$

where $\mu$ is a mutation magnitude hyperparameter. The SM-G-SUM operator [22] uses the gradient of attention weight of each node in a randomly sampled decoding step over a batch of instances to compute the sensitivity $\epsilon$, formally defined as follows,

$$\epsilon = \sqrt{\frac{1}{N} \sum_{k=0}^{N} \sum_{j=1}^{B} \nabla_{\theta} p_{th}(v_{t,j} = k|s_{t-1,j}, G_j)^2},$$

(21)

where $N$ is the number of nodes, $B$ is the batch size of the training instances $G_j$, and $t$ is the time step in the decoding process, which is randomly selected from $\{1, 2, ..., T\}$. $s_{t-1,j}$ is a partial solution at time step $t-1$ for the instance $G_j$, so it gives a random sample state of $G_j$.

The parent and offspring models are evaluated with a randomly sampled MO-VRPTW instance. Specifically, each model takes the instance as input and constructs solution for the instance. Then, a fitness vector (objective vector) is computed for the solution, and the model is marked with the fitness vector. After that, a multi-objective based selection is performed. Specifically, all the models are ranked according
to non-domination level and crowding distance of fitness vectors as in NSGA-II \cite{3}. After ranking, $M$ individuals (policies/models) are reserved. Algorithm 1 shows the procedure of evolutionary learning. The overview of hybrid training algorithm and multi-objective based selection are shown in Fig. 2.

D. The Procedure of MODRL/D-EL

Algorithm 2 shows the procedure of MODRL/D-EL. MODRL/D-EL first decomposes MO-VRPTW into $M$ single objective subproblems by weighted sum approach. For each subproblem, an attention model with comprehensive context information is trained by reinforcement learning with parameter-transfer strategy \cite{7}. Then, the trained models are taken as the initial population for evolutionary learning. The evolutionary learning uses proximal mutation \cite{21} to generate offsprings, and select individuals (policies/models) according to the ranks of non-domination level and crowding distance as in NSGA-II \cite{3}.

Algorithm 2 MODRL/D-EL

\textbf{Input:} The model set of subproblems $\Pi = \{\pi_1, \pi_2, ..., \pi_M\}$, weight vectors $\Lambda = \{\lambda_1, \lambda_2, ..., \lambda_M\}$, maximum iteration in evolutionary learning $D$  

\textbf{Output:} optimized model set $\Pi^* = \{\pi_1^*, \pi_2^*, ..., \pi_M^*\}$

\begin{enumerate}
\item randomly initialize parameters of models in $\Pi$
\item \textbf{for} $i = 1, ..., D$ \textbf{do}
\item train policy $\pi_i$ by DRL as described in Section 3.3 for $i$th subproblem
\item \textbf{end for}
\item take policy set trained by DRL as the initial population for evolutionary learning
\item \textbf{for} $i = 1, ..., M$ \textbf{do}
\item evolve individuals (policies/models) as described in Algorithm 1
\item \textbf{end for}
\end{enumerate}

V. EXPERIMENTAL RESULTS

All the experiments are executed on a machine with an Intel Xeon E5-2637 CPU and 16GB memory. All the related network models are trained on a single 1080Ti GPU.

A. Experiment Settings

1) Data Generation: The vehicles capacity $Q$ is set uniformly to 750, 900 and 1000 for problem size 50, 80 and 100, respectively. The time window of depot is set to $[a_0 = 0, b_0 = 10]$, and service duration $w$ of customers is set to 0.1 time unit. The coordinates of nodes, demands and time windows of customers are defined as follows \cite{20}:

- \textbf{Coordinates:} The coordinates of nodes are sampled uniformly from the interval $[0, 1]$.
- \textbf{Demands:} The demand $q_i$ is sampled from a normal distribution $\hat{q}_i \sim N(15,10)$, and truncated its absolute value to integer values between 1 and 42, i.e., $q_i^* = \min(42, \max(1, |\hat{q}_i|))$. Then, $q_i^*$ is scaled by the vehicle capacity $Q$, i.e. $q_i = q_i^*/Q$. The vehicle capacity $Q$ is also normalized to 1 in training.

- \textbf{Time Window:} For customer $v_i$, $h_i = [a_{\text{sample}} = h_i, b_{\text{sample}} = b_0 - h_i - w]$ is defined, where $h_i = d_i/0.01$, and $d_i$ is the traveling time from $v_i$ to the depot. Then, the start time $a_i$ is sampled uniformly from $h_i$, and the end time $b_i$ is calculated as $b_i = \min(a_i + 3\alpha, b_{\text{sample}})$, $\alpha = \max(|\hat{a}_i|, 0.01)$, $\hat{a} \sim N(0,1)$.

2) Training Details: In deep reinforcement learning, the subproblem number $M$ is set to 100. The weight vectors for subproblems are uniformly spread from 0 to 1, i.e. $\lambda_1 = (0, 1), \lambda_2 = (0.01, 0.99), ..., \lambda_{100} = (0.99, 0.01)$. The value function in REINFORCE algorithm \cite{19} is the greedy rollout baseline proposed in \cite{13}. Adam \cite{23} is used as optimizer, and the learning rate is set to $10^{-4}$ constantly. The number of heads in MHA is set to 8. The dimension $d_{\text{emb}}$ of $h_{\text{node}}$ and $h_{\text{vehicle}}$ are set to 128. The hidden dimension in FF layer is 256. The hidden dimension of $g_t$ and $g_t$ is set to 64.

Based on the neighborhood-based parameter-transfer strategy \cite{17}, the first model is trained for 10 epochs, each epoch involving 512,000 instances. The remaining 99 models are only trained 1 epoch; each epoch contains 256,000 instances. The training batch size in DRL is set to 128.

In evolutionary learning, the hyperparameter $\mu$ in proximal mutation is set to 0.01. The training batch size in evolutionary learning is set to 64. The maximum generation $D$ is set to 30.

The training time for our models is about 20 hours.

3) Baselines: Two learning-based algorithms are introduced as competitors to evaluate the performance of MODRL/D-EL. The first one is MODRL/D-AM \cite{14}, and the second one is MODRL/D-CCI, which extends MODRL/D-AM by integrating the comprehensive context information. MODRL/D-CCI is compared with MODRL/D-AM to evaluate the contribution of CCI. MODRL/D-EL is compared with MODRL/D-CCI to evaluate the contribution of evolutionary learning.

MODRL/D-EL is also compared with three classical iterative based algorithms: NSGA-II, MOEA/D and MOGLS. For NSGA-II and MOEA/D, the number of iterations is set to 20000. The population size is set to 100. For MOGLS, the size of the temporary population and the number of the initial solutions are consistent with the setting in \cite{3}. The local search uses 2-opt operator, which is terminated when 100 iterations are reached. The number of iterations in MOGLS is set to 4000 for a similar testing time with NSGA-II.

B. Results and Discussions

All the attention models used in learning-based algorithms are trained with the instances of size 50. The average performance over a batch of instances can evaluate the algorithms more accurately. The learning-based algorithms learn from the training data and thus, they can efficiently construct solutions for a batch of instances from the same distribution. The iterative-based algorithms do not require the tested instances to follow some distribution. However, the iteration process needs
to be re-performed for every run. Hence, a tremendous amount of time is required to solve a batch of instances.

The experiments consist of two parts. For the first part, to compare MODRL/D-EL with other learning-based algorithms, three testing sets of problem size 50, 80 and 100 are respectively sampled from the same distribution as training data. Each testing set contains 1024 instances. The iterative-based algorithms are not tested in this part, because these algorithms take a tremendous amount of time to solve 1024 instances. For the second part, to compare MODRL/D-EL with iterative-based algorithms, 11 instances (R201–R211) from Solomon benchmark [15] are used as testing instances. Hypervolume (HV) and the number of solutions in non-dominant set (NDS) are taken as performance indicators. The reference point for calculating HV are set to (100,10), (120,10) and (160,10) for problem size 50, 80 and 100 respectively.

Table II shows the experimental results on the three testing sets. In comparisons between MODRL/D-CCI and MODRL/D-AM, MODRL/D-CCI achieves better performance in term of HV on all the three testing sets, which suggests that comprehensive context information is helpful to make more reasonable decision. In comparisons between MODRL/D-EL and MODRL/D-CCI, MODRL/D-EL achieves better performance in term of HV on the testing sets of problem size 80 and 100. In addition, MODRL/D-EL obtains more non-dominant solutions on all the three testing sets. The fact suggests that evolutionary learning can fine-tune the parameters of models to improve the quality of solutions. The testing time of MODRL/D-EL is a little longer than that of MODRL/D-AM. This is because processing comprehensive context information requires more testing time than processing simple context information. To further visualize the final solutions found by different approaches, Fig. 2 shows the approximate Pareto fronts of two instances with different problem size. As shown in the figure, MODRL/D-EL has a better performance in both convergence and diversity.

Table III reports the experimental results on 11 benchmark instances R201–211. The mean values and standard deviation (STD) are shown in the bottom. To eliminate the impact of randomness in NSGA-II, MOEA/D and MOGLS, these algorithms are run 5 times independently and the mean values are taken as final results. The highest values of HV are highlighted in bold.

VI. CONCLUSION

This paper proposes a multiobjective deep reinforcement learning and evolutionary learning algorithm for MOPs with complex constraints. In the proposed algorithm, the attention model constructs solution by using more comprehensive context information of MOP, which is helpful in making reasonable decision. The evolutionary learning is employed in the proposed algorithm to fine-tune the parameters of models to achieve better performance. The proposed algorithm is compared with other learning-based and iterative-based algorithms for MO-VRPTW. The experimental results suggest that the proposed algorithm has a better performance in both convergence and diversity.
With respect to the future studies, the evolutionary learning in this paper only considers mutation operator. A crossover operator may generate offspring models that construct solutions with higher diversity and convergence. Besides, the MO-VRPTW instances in this paper are symmetrical. How to efficiently deal with asymmetric instances is also worth further studying.

REFERENCES

[1] Q. Zhang and H. Li, “MOEA/D: A multiobjective evolutionary algorithm based on decomposition,” IEEE Transactions on Evolutionary Computation, vol. 11, no. 6, pp. 712–731, 2007.

[2] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, “Multiobjective evolutionary algorithms: A survey of the state of the art,” Swarm and Evolutionary Computation, vol. 1, no. 1, pp. 32–49, 2011.

[3] K. Li, T. Zhang, and R. Wang, “Deep reinforcement learning for multiobjective optimization,” IEEE Transactions on Cybernetics, pp. 1–12, 2020.

[4] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takáč, “Reinforcement learning for solving the vehicle routing problem,” in NeurIPS, 2018, pp. 9839–9849.

[5] J. Castro-Gutierrez, D. Landa-Silva, and J. M. Perez, "Nature of real-world multi-objective vehicle routing with evolutionary algorithms," in IEEE International Conference on Systems, Man, and Cybernetics, 2011, pp. 257–264.

[6] K. Miettinen, Nonlinear multiobjective optimization. Springer Science & Business Media, 2012, vol. 12.

[7] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in CVPR, 2016, pp. 770–778.

[8] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in ICML, 2015, pp. 448–456.

[9] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” Machine Learning, vol. 8, no. 3, pp. 229–256, 1992.

[10] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takáč, “Reinforcement learning for solving the vehicle routing problem,” in NeurIPS, 2018, pp. 9839–9849.

[11] J. Miettinen, Nonlinear multiobjective optimization. Springer Science & Business Media, 2012, vol. 12.

[12] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in CVPR, 2016, pp. 770–778.

[13] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in ICML, 2015, pp. 448–456.

[14] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” Machine Learning, vol. 8, no. 3, pp. 229–256, 1992.

[15] 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.

[16] C. Bodnar, B. Day, and P. Lió, “Proximal distilled evolutionary reinforcement learning,” in AAAI, vol. 34, no. 4, 2020, pp. 3283–3290.

[17] J. Lehman, J. Chen, J. Clune, and K. Stanley, “Safe mutations for deep and recurrent neural networks through output gradients,” in GECCO, 2018, pp. 117–124.

[18] D. P. Kingma and J. L. Ba, “Adam: A method for stochastic optimization,” in ICLR, 2015.