Deep Reinforcement Learning for Multi-objective Optimization

Kaiwen Li, Tao Zhang, and Rui Wang

***Abstract*—This study proposes an end-to-end framework for solving multi-objective optimization problems (MOPs) using Deep Reinforcement Learning (DRL), termed DRL-MOA. The idea of decomposition is adopted to decompose a MOP into a set of scalar optimization subproblems. The subproblems are then optimized cooperatively by a neighbourhood-based parameter transfer strategy which significantly accelerates the training procedure and makes the realization of DRL-MOA possible. The subproblems are modelled as neural networks and the RL method is used to optimize them. In specific, the multi-objective travelling salesman problem (MOTSP) is solved in this work using the DRL-MOA framework by modelling the subproblem as the Pointer Network. It is found that, once the trained model is available, it can scale to MOTSPs of any number of cities, e.g., 70-city, 100-city, even the 200-city MOTSP, without re-training the model. The Pareto Front can be directly obtained by a simple feed-forward of the network; thereby, no iteration is required and the MOP can be always solved in a reasonable time. Experimental results indicate a strong convergence ability of the DRL-MOA, especially for large-scale MOTSPs, e.g., 200-city MOTSP, for which evolutionary algorithms such as NSGA-II and MOEA/D are pretty hard to converge even implemented for a large number of iterations. The DRL-MOA can also obtain a much wider spread of the PF than the two competitors. Moreover, the DRL-MOA has a high level of modularity and can be easily generalized to other MOPs by replacing the modelling of the subproblem.**

***Index Terms*—Multi-objective optimization, Reinforcement learning, Travelling salesman problem, Decomposition, Evolutionary algorithm.**

I. Introduction

MULTI-OBJECTIVE optimization problems arise regularly in real-world where two or more objectives are required to be optimized simultaneously. Without loss of generality, a MOP can be defined as follows:

$$\begin{aligned} \min_{\mathbf{x}} \ & \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_M(\mathbf{x})) \\ \mathbf{s.t.} \ & \mathbf{x} \in X, \end{aligned}$$

where $\mathbf{f}(\mathbf{x})$ is consisted of M different objective functions and $X \subseteq R_D$ is the decision space. Since the M objectives are usually conflicting with each other, a set of trade-off solutions, termed Pareto optimal solutions, are expected to be found for MOPs.

This paper is partially supported by the National Natural Science Foundation of China (No. 61773390 and No. 71571187).

Kaiwen Li, Rui Wang (corresponding author), and Tao Zhang was with the College of Systems Engineering, National University of Defense Technology, Changsha 410073, PR China. (e-mail: kaiwenli_nudt@foxmail.com, ruiwangnudt@gmail.com, zhangtao@nudt.edu.cn)

Manuscript received May 19, 2019; revised August 26, 2019.

Among MOPs, various multi-objective combinatorial optimization problems have been investigated in recent years. A canonical example is the multi-objective travelling salesman problem (MOTSP), where given n cities and p costs to travel from city i to j, one needs to find a cyclic tour of the n cities, minimizing the p cost functions. This is a NP-hard problem even for the single-objective TSP. The best known exact method, i.e., dynamic programming algorithm, requires a complexity of $\Theta\left(2^n n^2\right)$ for single-objective TSP. Also, it appears to be harder for its multi-objective version. Hence, in practice, approximate algorithms are commonly used to solve MOTSPs, i.e., finding near optimal solutions.

During the last two decades, multi-objective evolutionary algorithms (MOEAs) have proven effective in dealing with MOPs since they can obtain a set of solutions in a single run due to their population based characteristic. NSGA-II [1] and MOEA/D [2] are two of the most popular MOEAs which have been widely studied and applied in many real world applications. The two algorithms as well as their variants have also been applied to solve the MOTSP, see e.g., [3], [4], [5].

In addition, several handcrafted heuristics especially designed according to the characteristics of TSP have been studied, such as the Lin-Kernighan heuristic [6] and the 2-opt local search [7]. By adopting these carefully designed tricks, a number of methods have been proposed to solve MOTSP, such as the Pareto local search method (PLS) [8] and the multiple objective genetic local search (MOGLS) method [9]. Other variants and more details of such methods can be found in [10].

It has been a long time that evolutionary algorithms and/or handcrafted heuristics are recognized as suitable to handle such problem. However, these algorithms, as iteration-based solvers, have suffered obvious limitations that have been widely discussed [11], [12], [10]. First, to find the near-optimal solution, especially when the dimension of problems is large, a large number of iterations are required for population updating or iterative searching, thus usually leading to a long computing time for optimization. Second, once there is a slight change of the problem, e.g., changing the coordinates of a city for MOTSP, the algorithm may need to be re-performed to compute the solutions. When it comes to newly encountered problems, or even new instances of a similar problem, the algorithm needs to be revised to obtain a good result, which is known as the *No Free Lunch theorem* [13]. Furthermore, such problem specific methods are usually optimized for one task only.

Carefully handcrafted evolution strategies and heuristics can certainly improve the performance. However, the recent

advances in machine learning algorithms have shown their ability of replacing humans as the engineers of algorithms to solve different problems. Several years ago, most people used man-engineered features in the field of computer vision but now the Deep Neural Networks (DNNs) have been the main techniques. While DNNs focus on making *predictions*, Deep Reinforcement Learning (DRL) is mainly used to learn how to make *decisions*. Thereby, we believe that DRL is a possible way of learning how to solve various optimization problems automatically, thus demanding no man-engineered evolution strategies and heuristics. In this work, we explore the possibility of using DRL to solve the multi-objective problem, MOTSP in specific, in an *end-to-end* manner, i.e., given n cities as input, the optimal solutions can be *directly* obtained by forwarding the trained neural network. The network model is trained through the trail and error process of DRL and can be viewed as a black-box heuristic or a meta-algorithm [14] with strong learned heuristics. Because of the exploring characteristic of DRL training, the obtained model shows a high level of generalization. Once the model is trained, it can solve a wide range of problems, e.g., any number of cities and arbitrary city coordinates, without re-training for new instances.

This work is originally motivated by several recent proposed Neural Network-based single-objective TSP solvers. [15] first proposes a Pointer Network that uses attention mechanism [16] to predict the city permutation. This model is trained in a supervised way that requires enormous TSP examples and their optimal tours as training set. It is hard for use and the supervised training process prevents the model from obtaining better tours than the ones provided in the training set. To resolve this issue, [17] adopts an Actor-Critic DRL training algorithm to train the Point Network with no need of providing the optimal tours. [14] simplifies the Point Network model and adds dynamic elements input to extend the model to solve the Vehicle Routing Problem (VRP). The recent progress in using DRL algorithms to solve the TSP is really appealing and inspiring due to its non-iterative yet efficient characteristic and high level of generalization. However, there are no such studies concerning solving MOPs (or the MOTSP in specific) by DRL based methods.

This study, therefore, proposes a DRL-based multi-objective optimization algorithm (DRL-MOA) to handle MOPs in a non-iterative manner with high generalization ability. The MOTSP is taken as a specific test problem. In the DRL-MOA first the decomposition strategy [2] is adopted to decompose MOTSP into a number of scalar optimization subproblems. A modified Pointer Network similar to [14] is used to model the subproblem and the Actor-Critic algorithm [18] is used for training. In particular, a neighborhood-based parameter sharing strategy is proposed to significantly accelerate the training procedure and improve the convergence.

This DRL-MOA framework is attractive for its self-driven learning mechanism that only requires the reward functions without any need of other information; the model explores and learns strong heuristics automatically in an unsupervised way. Then the trained model gains the capability to solve MOTSP with a high generalization ability. With a slight change of the problem instance, e.g., changing the number or coordinates of the cities, existing heuristic methods require to be re-conducted from scratch, which is usually impractical for application, especially when the problem dimension is large. In contrast, the proposed DRL-MOA is robust to the problem perturbation and is able to obtain the near-optimal solutions given any number of cities and arbitrary city coordinates, with no need of re-training the model. In addition, this framework solves the MOTSP in a non-iterative manner, that is, a set of Pareto optimal solutions can be directly obtained by a feed-forward pass of the trained network without any population updating or searching iteration procedure. This feature overcomes the underlying limitation of existing iterative heuristic methods, i.e., the long computing time due to the large number of iterations.

II. The Reinforcement learning-based multi-objective optimization algorithm (DRL-MOA)

We first introduce the general framework of DRL-MOA, where decomposition strategy and neighborhood-based parameter transfer strategy are used together to solve the MOPs. Then the MOTSP is taken as a specific test problem to elaborate how to model and solve the MOTSP using the proposed DRL-MOA.

A. General framework

Decomposition strategy. Decomposition, as a simple yet efficient way to design the multi-objective optimization algorithms, has fostered a number of researches in the community, e.g., MOEA/D, MOEA/DD [19] and NSGA-III [20]. The idea of decomposition is also adopted as the basic framework of the proposed DRL-MOA in this work. The MOP, e.g., the MOTSP, is explicitly decomposed into a set of scalar optimization subproblems and solved in a collaborative manner. Each solution is associated with a scalar optimization problem. The desired Pareto Front (PF) can be obtained when all the scalar optimization problems are solved.

In specific, the well-known Weighted Sum [21] approach is employed. Certainly, other scalarizing methods can also be applied, e.g., the Chebyshev and the penalty boundary intersection (PBI) method [22], [23]. First a set of uniformly spread weight vectors $\lambda^1, \ldots, \lambda^N$ is given, e.g., $(1,0), (0.9,0.1), \ldots, (0,1)$ for a bi-objective problem, as shown in Fig. 1. Here $\lambda^j = (\lambda_1^j, \ldots, \lambda_m^j)^T$ and m represents the number of objectives. Thus the problem of approximating the PF is converted into N scalar optimization subproblems by the Weighted Sum approach. The objective function of the j_{th} subproblem is shown as follows [2]:

$$\text{minimize } g^{ws}(x|\lambda_i^j) = \sum_{i=1}^{m} \lambda_i^j f_i(x) \tag{1}$$

Therefore, the PF is finally formed by the solutions obtained by solving all the N subproblems.

Neighborhood-based parameter transfer strategy. The N scalar optimization subproblems are solved in a collaborative manner by the neighborhood-based parameter transfer

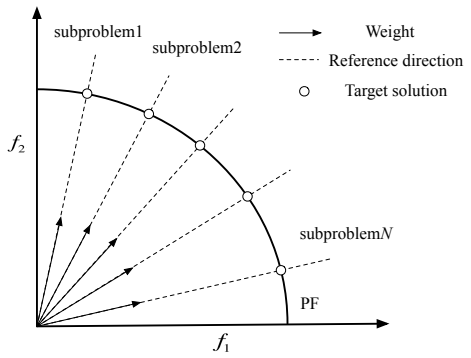


Fig. 1. Illustration of the decomposition strategy.

strategy. According to Eq. (1) it can be observed that two neighbouring subproblems could have very close optimal solutions [2]. Thus, a subproblem can be solved assisted by the information of its neighboring subproblems.

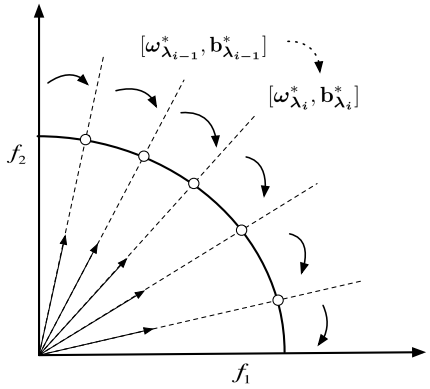


Fig. 2. Illustration of the parameter-transfer strategy.

Specifically, as the subproblem in this work is modelled as a neural network, the parameters of the $(i-1)_{th}$ subproblem can be expressed as $[\boldsymbol{\omega}^*_{\boldsymbol{\lambda_{i-1}}}, \mathbf{b}^*_{\boldsymbol{\lambda_{i-1}}}]$. Assume that this subproblem has been solved, i.e., the parameters have been optimized. Then the best parameters $[\boldsymbol{\omega}^*_{\boldsymbol{\lambda_{i-1}}}, \mathbf{b}^*_{\boldsymbol{\lambda_{i-1}}}]$ obtained in the $(i-1)_{th}$ subproblem are set as the starting point for the network training in the i_{th} subproblem. Briefly, the network parameters are transferred from the previous subproblem to the next subproblem in a sequence, as depicted in Fig. 2. The neighborhood-based parameter transfer strategy makes it possible for the training of the DRL-MOA model; otherwise a tremendous amount of time is required for training the N subproblems.

Each subproblem is modelled and solved by the DRL algorithm and all subproblems can be solved in sequence based on the parameter transferring. Thus, the PF can be finally approximated according to the obtained model. Employing the decomposition in conjunction with the neighborhood-based parameter transfer strategy, the general framework of DRL-MOA is presented in Algorithm 1.

One obvious advantage of the DRL-MOA is its modularity. For example, based on this framework, the MOTSP can be solved efficiently by integrating any of the recently proposed novel DRL-based TSP solvers. Also, other problems beside of the TSP, such as VRP, can be easily handled with the DRL-MOA framework by replacing the model of the subproblem. Once the trained model is available, the PF can be directly obtained by a simple feed-forward calculation of the model. Importantly, the trained model can adapt to any change of the problem, as long as the problem settings are generated from the same distribution with the training set, e.g., the city coordinates of training set and test problems are both sampled from [0,1] uniformly.

Algorithm 1 General Framework of DRL-MOA

Input: The model of subproblem $\mathcal{M} = [\boldsymbol{w}, \mathbf{b}]$, weight vectors $\lambda^1, \ldots, \lambda^N$
Output: The optimal model $\mathcal{M}^* = [\boldsymbol{w}^*, \mathbf{b}^*]$

```
[ω_λ1, b_λ1] ← Random_Initialize
for i ← 1 : N do
    if i == 1 then
        [ω*_λ1, b*_λ1] ← Actor_Critic([ω_λ1, b_λ1], g^ws(λ1))
    else
        [ω_λi, b_λi] ← [ω*_λi−1, b*_λi−1]
        [ω*_λi, b*_λi] ← Actor_Critic([ω_λi, b_λi], g^ws(λi))
    end if
end for
return [w*, b*]
```

Given inputs of the MOP, the PF can be directly calculated by $[\boldsymbol{w}^*, \mathbf{b}^*]$.

B. Modelling the subproblem of MOTSP

Based on the foregoing DRL-MOA framework, this section solves the MOTSP by introducing the modelling of the subproblem of MOTSP. A modified Pointer Network similar to [14] is used to model the subproblem and the Actor-Critic algorithm is used for training.

1) The model: We first introduce how to model the subproblem of MOTSP. It is noted that the subproblem of MOTSP is not the same as the traditional TSP due to its multiple inputs beside of the city coordinates and its Weighted-sum-based reward evaluation.

More formally, let the given set of inputs be $X \doteq \{x^i, i = 1, \cdots, n\}$ where n is the number of cities. Each x^i is represented by a tuple $\{x^i = (x^i_1, \cdots, x^i_M)\}$ where M is the number of objectives. Taking the bi-objective TSP as an example, x^i_1 represents the coordinates of the i_{th} city and x^i_2 represents the second input, e.g., the security index, of the i_{th} city. Thus the goal is to find a permutation of the cities $Y = \{y_1, \cdots, y_n\}$, termed a cyclic tour, to minimize the aggregated objective functions. First an arbitrary city is selected as y_1. At each decoding step $t = 1, 2, \cdots$, we choose y_{t+1} from the available cities X_t. The available cities X_t are updated every time a city has been visited. This process is modelled using the probability chain rule:

$$P(Y|X) = \prod_{t=1}^{n} P(y_{t+1}|y_1, \cdots, y_t, X_t) \tag{2}$$

In a nutshell, Eq. (2) provides the probability of selecting the next city according to $y_1, \cdots, y_t$. Here, a modified Pointer

network similar to [14] is used to compute the conditional probability of Eq. (2). Its basic structure is the Sequence-to-Sequence model [24], a recently proposed powerful model in the field of machine translation, which maps one sequence to another. The general Sequence-to-Sequence model consists of two RNN networks, termed encoder and decoder. An encoder RNN encodes the input sequence into a code vector that contains knowledge of the input. Based on the code vector, a decoder RNN is used to decode the knowledge vector to a desired sequence.

In this work, the architecture of the model is shown in Fig. 3 where the left part is the encoder and the right part is the decoder. The model is elaborated as follows.

Encoder. Since the coordinates of the cities convey no sequential information [14] and the order of city locations in the inputs is not meaningful, RNN is not used in the encoder in this work. Instead, a simple embedding layer is used to encode the inputs to a code vector which can decrease the complexity of the model and reduce the computational cost. Specifically, the 1-dimensional (1-D) convolution layer is used to encode the inputs to a high-dimensional vector space [14]. The number of in-channels equals to the dimension of the inputs. For instance, if both the cost functions of the bi-objective TSP are defined by the Euclidean distance between two points, the number of in-channels is four, since two inputs are required to calculate the Euclidean distance. It is noteworthy that the parameters of the 1-D convolution layer are shared amongst all the cities. Thus, the encoder is robust to the number of the cities.

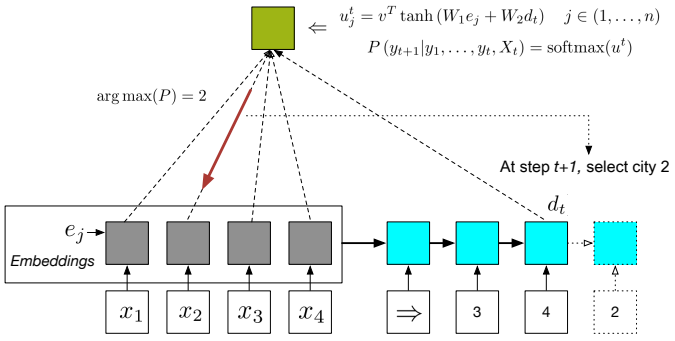


Fig. 3. Illustration of the attention mechanism. Attention mechanism produces the probability of selecting the next city

Decoder. Different from the encoder, a RNN is required in the decoder as we need to summarize the information of previous steps $y_1, \cdots, y_t$ so as to make the decision of y_{t+1}. RNN has the ability of memorizing the previous outputs. In this work we adopt the RNN model of GRU (Gated recurrent unit) [25] that has similar performance but fewer parameters than the LSTM (Long Short-Term Memory) which is employed in the original Pointer Network in [14]. It is noted that RNN is not directly used to output the sequence. What we need is the RNN decoder hidden state d_t at decoding step t that stores the knowledge of previous steps $y_1, \cdots, y_t$. Then d_t and the encoding of the inputs $e_1, \cdots, e_n$ are used together to calculate the conditional probability $P(y_{t+1}|y_1, \ldots, y_t, X_t)$ over the next step of city selection. This calculation is realized by the attention mechanism.

Attention mechanism. Intuitively, the attention mechanism calculates how much every input is relevant in the next decoding step t. The most relevant one is given more *attention* and can be selected as the next visiting city. The calculation is as follows:

$$\begin{gathered} u_j^t = v^T \tanh(W_1 e_j + W_2 d_t) \quad j \in (1,\ldots,n) \\ P(y_{t+1}|y_1,\ldots,y_t,X_t) = \text{softmax}(u^t) \end{gathered} \tag{3}$$

where v, W_1, W_2 are learnable parameters. d_t is a key variable for calculating $P(y_{t+1}|y_1, \ldots, y_t, X_t)$ as it stores the information of previous steps $y_1, \cdots, y_t$. Then, for each city j, its u_j^t is computed by d_t and its encoder hidden state e_j, as shown in Fig. 3. The softmax operator is used to normalize $u_1^t, \cdots, u_n^t$ and finally the probability for selecting each city j at step t can be finally obtained. The greedy decoder can be used to select the next city. For example, in Fig. 3, city 2 has the largest $P(y_{t+1}|y_1, \ldots, y_t, X_t)$ and so is selected as the next visiting city.

2) Training method: The model of the subproblem is trained using the well-known Actor-critic method similar to [17], [14]. However, as [17], [14] trains the model of single-objective TSP, the training procedure is different for the MOTSP case, as presented in Algorithm 2. Next we briefly introduce the training procedure.

Two networks are required for training: (i) an actor network, which is exactly the Pointer Network in this work, gives the probability distribution for choosing the next action, and (ii) a critic network that evaluates the expected reward given a specific problem sate. The critic network employs the same architecture as the pointer networks encoder. Then two fully connected layers map the encoder hidden state into the critic output.

The training is conducted in an unsupervised way. During the training, we generate the MOTSP instances from distributions $\{\Phi_{\mathcal{M}_1}, \cdots, \Phi_{\mathcal{M}_M}\}$. Here, $\mathcal{M}$ represents different input features of the cities, e.g., the city locations or the security indices of the cities. M is the number of objectives. For example, for Euclidean instances of a bi-objective MOTSP, $\mathcal{M}_1$ and $\mathcal{M}_2$ are both city coordinates and $\Phi_{\mathcal{M}_1}$ or $\Phi_{\mathcal{M}_2}$ can be a uniform distribution of $[0, 1] \times [0, 1]$.

To train the actor and critic networks with parameters θ and ϕ, N instances are sampled from $\{\Phi_{\mathcal{M}_1}, \cdots, \Phi_{\mathcal{M}_M}\}$ for training. For each instance, we use the actor network with current parameters θ to produce the cyclic tour of the cities and the corresponding reward can be computed. Then the policy gradient is computed in step 11 (refer to [26] for details of the formula derivation of policy gradient) to update the actor network. Here, $V(X_0^n; \phi)$ is the reward approximation of instance n calculated by the critic network. The critic network is then updated in step 12 by reducing the difference between the true observed rewards and the approximated rewards.

III. Experiment

A. Experimental description

In this work, we test our method on bi-objective TSPs. Particularly, two types of bi-objective TSP are considered [10]:

Algorithm 2 Actor-Critic training algorithm

Input: $\theta, \phi \leftarrow$ initialized parameters given in Algorithm 1
Output: The optimal parameters θ, ϕ

for $iteration \leftarrow 1, 2, \cdots$ **do**
  generate N problem instances from $\{\Phi_{\mathcal{M}_1}, \cdots, \Phi_{\mathcal{M}_M}\}$ for the MOTSP.
  for $k \leftarrow 1, \cdots, N$ **do**
    $t \leftarrow 0$
    while not terminated **do**
      select the next city y_{t+1}^k according to $P\left(y_{t+1}^k | y_1^k, \cdots, y_t^k, X_t^k\right)$
      Update X_t^k to X_{t+1}^k by leaving out the visited cities.
    end while
    compute the reward R^k
  end for
  $d\theta \leftarrow \frac{1}{N} \sum_{k=1}^{N} \left(R^k - V\left(X_0^k; \phi\right)\right) \nabla_\theta \log P\left(Y^k | X_0^k\right)$
  $d\phi \leftarrow \frac{1}{N} \sum_{k=1}^{N} \nabla_\phi \left(R^k - V\left(X_0^k; \phi\right)\right)^2$
  $\theta \leftarrow \theta + \eta d\theta$
  $\phi \leftarrow \phi + \eta d\phi$
end for

- Euclidean instances: both the cost functions are defined by the Euclidean distance. The first cost is defined by the Euclidean distance between the real coordinates of two cities i, j. The second cost of travelling from city i to city j is defined by another set of *virtual* coordinates, e.g., the Euclidean distance between randomly generated $(0.2, 0.7)$ and $(0.3, 0.5)$.
- Mixed instances: the first cost function is defined by the Euclidean distance between two points. The second cost of travelling from city i to j is a random value uniformly sampled from $[0, 1]$.

The two types of bi-objective TSP have different problem structures and thus require different model structures. For Mixed instances, the dimension of input is three because a city coordinate (x, y) and a random value are required. However, four inputs are needed for Euclidean instances as two sets of city coordinates are required for the calculation of the two cost functions.

In addition, to find out whether the number of cities of the training set would influence the DRL-MOA performance, we train the model using instances of 20 cities and 40 cities, respectively. Thus, in total four models are trained based on the four problem settings of training, namely, Euclidean 20-city instances, Euclidean 40-city instances, Mixed 20-city instances, Mixed 40-city instances.

To evaluate the models, bi-objective TSP of 40 cities, 70 cities, 100 cities, 150 cities, 200 cities are tested on the trained model. In specific, to evaluate the Euclidean bi-objective TSP, the standard TSP test problems kroA and kroB in the TSPLIB library [27] are used to construct the Euclidean test instances kroAB100, kroAB150 and kroAB200. kroA and kroB are two sets of different city locations. Here, kroA and kroB are set as two inputs to calculate the two Euclidean costs. For Mixed test instances, the three inputs are generated randomly from $[0, 1]$.

B. Parameter settings of model and training

Most parameters of model and training are similar to that in [14] which solves the single-objective TSP effectively. Specifically, the parameter settings of the network model are shown in TABLE I. D_{input} represents the dimension of input, i.e., $D_{input} = 4$ for Euclidean bi-objective TSP. We employ an one-layer GRU RNN with the hidden size of 128 in the decoder. For the critic network, the hidden size is also set to 128.

TABLE I
PARAMETER SETTINGS OF THE MODEL. 1D-CONV MEANS THE 1-D CONVOLUTION LAYER. D_{input} REPRESENTS THE DIMENSION OF INPUT. KERNEL SIZE AND STRIDE ARE ESSENTIAL PARAMETERS OF THE 1-D CONVOLUTION LAYER

Actor network(Pointer Network)	
Encoder:	1D-Conv(D_{input}, 128, kernel size=1, stride=1)
Decoder:	GRU(hidden size=128, number of layer=1)
	Attention(No hyper parameters)
Critic network	
1D-Conv(D_{input}, 128, kernel size=1, stride =1)	
1D-Conv(128, 20, kernel size=1, stride =1)	
1D-Conv(20, 20, kernel size=1, stride =1)	
1D-Conv(20, 1, kernel size=1, stride =1)	

We train both of the actor and critic networks using the Adam optimizer [28] with learning rate η of 0.0001 and batch size of 200. The Xavier initialization method [29] is used to initialize the weights for the first subproblem. Weights for the following subproblems are generated by the introduced neighborhood-based parameter transfer strategy.

In addition, different size of generated instances are required for training different types of models. As compared with the Mixed MOTSP problem, the model of Euclidean MOTSP problem requires more weights to be optimized because its dimension of input is larger, thus requiring more training instances in each iteration. In this work, we generate 500,000 instances for training the Euclidean bi-objective TSP and 120,000 instances for training the Mixed one. All the problem instances are generated from a uniform distribution of $[0, 1]$ and used in training for 5 epoches. The above settings are roughly determined by experiments.

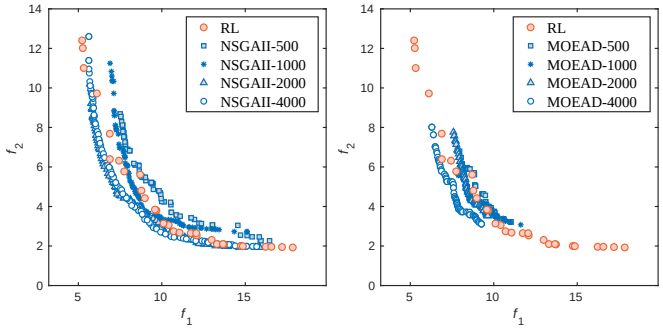
C. Results and discussions

We compare the PF found by the DRL-MOA with those obtained by NSGA-II and MOEA/D algorithms. The maximum number of iteration for NSGA-II and MOEA/D is set to 500, 1000, 2000 and 4000 respectively. The population size is set to 100 for NSGA-II and MOEA/D. The number of subproblems for DRL-MOA is set to 100 as well. In addition, only the non-dominated solutions are reserved in the final PF.

1) Experiment of Mixed type bi-objective TSP: We first test the model that is trained on 40-city Mixed type bi-objective TSP instances. The model is then used to approximate the PF of 40-, 70-, 100-, 150- and 200-city problems.

Fig. 4, 5, 6, 7, 8 show the results of solving 40-, 70-, 100-, 150- and 200-city problems. It is obvious that, once the model is trained, it can be directly used to solve bi-objective TSP with different number of cities. Although the model is obtained by training the 40-city TSP problem, it can still perform efficiently on the 70-, 100-, 150- and 200-city problems. The performance indicator of Hypervolume (HV) and the computing time for the above methods are also listed in Table II.

As shown in Fig. 4, for the bi-objective TSP with a small number of cities like 40, all the methods, i.e., NSGA-II, MOEA/D and the DRL-MOA can work well. By increasing the number of iterations, NSGA-II and MOEA/D even show a better ability of convergence. However, the large number of iterations can lead to a large amount of computing time. For example, 4000 iterations cost 130.2 seconds for MOEA/D and 28.3 seconds for NSGA-II while our method just requires 2.7 seconds.

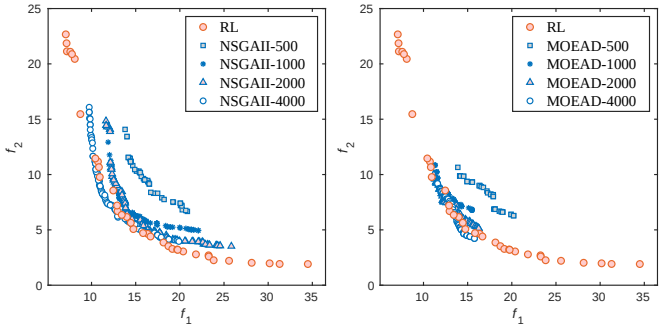


(a) DRL-MOA and NSGA-II (b) DRL-MOA and MOEA/D

Fig. 4. A random generated **40-city** Mixed bi-objective TSP problem instance: the PF obtained using our method (trained using 40-city instances) in comparison with NSGA-II and MOEA/D. 500, 1000, 2000, 4000 iterations are applied respectively.

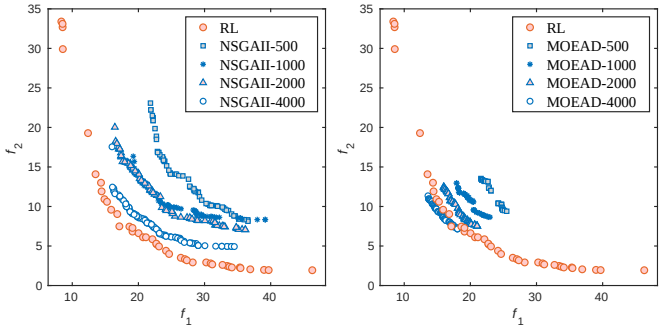
As can be seen in Fig. 6, 7, 8, as the number of cities increases, both NSGA-II and MOEA/D struggle to converge while the DRL-MOA exhibits a significantly enhanced ability of convergence. For 100-city problems in Fig. 6, MOEA/D shows a slightly better performance in terms of convergence than other methods by running 4000 iterations with 140.3 seconds. However, the diversity of solutions found by our method is much better than MOEA/D. For 150- and 200-city problems as depicted in Fig. 7 and Fig. 8, NSGA-II and MOEA/D exhibit an obviously inferior performance than our method in terms of both the convergence and diversity. Even though NSGA-II and MOEA/D are conducted for 4000 iterations, which effectively is a pretty large number of iterations, DRL-MOA still shows a much better performance than them.

In addition, the DRL-MOA achieves the best HV comparing to other algorithms, as shown in TABLE II. Also, its computing time is reasonable in comparison with NSGA-II and MOEA/D. Overall, from the above results, we can clearly



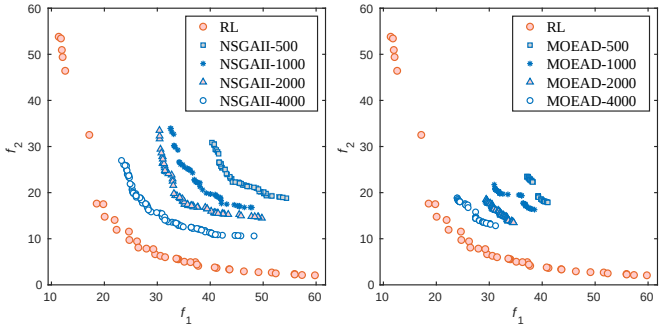
(a) DRL-MOA and NSGA-II (b) DRL-MOA and MOEA/D

Fig. 5. A random generated **70-city** Mixed bi-objective TSP problem instance: the PF obtained using our method (trained using 40-city instances) in comparison with NSGA-II and MOEA/D. 500, 1000, 2000, 4000 iterations are applied respectively.



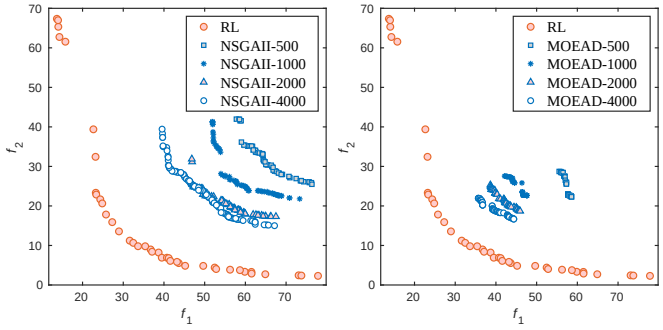
(a) DRL-MOA and NSGA-II (b) DRL-MOA and MOEA/D

Fig. 6. A random generated **100-city** Mixed bi-objective TSP problem instance: the PF obtained using our method (trained using 40-city instances) in comparison with NSGA-II and MOEA/D. 500, 1000, 2000, 4000 iterations are applied respectively.



(a) DRL-MOA and NSGA-II (b) DRL-MOA and MOEA/D

Fig. 7. A random generated **150-city** Mixed bi-objective TSP problem instance: the PF obtained using our method (trained using 40-city instances) in comparison with NSGA-II and MOEA/D. 500, 1000, 2000, 4000 iterations are applied respectively.



(a) DRL-MOA and NSGA-II (b) DRL-MOA and MOEA/D

Fig. 8. A random generated **200-city** Mixed bi-objective TSP problem instance: the PF obtained using our method (trained using 40-city instances) in comparison with NSGA-II and MOEA/D. 500, 1000, 2000, 4000 iterations are applied respectively.

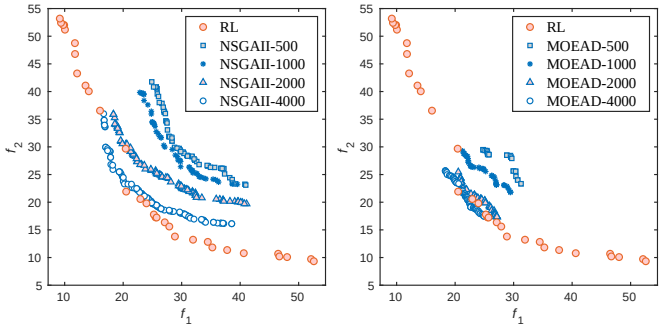
observe the enhanced ability of DRL-MOA on solving large-scale bi-objective TSPs. The *brain* of the trained model has learned how to select the next city given the city information and the selected cities. Thus it does not suffer the deterioration of performance with the increasing number of cities. In contrast, NSGA-II and MOEA/D fail to converge within a reasonable computing time for large-scale bi-objective TSPs. Moreover, the PF obtained by the DRL-MOA framework shows a significantly better diversity as compared with NSGA-II and MOEA/D whose PF has a much smaller spread.

2) Experiment of Euclidean type bi-objective TSP: We then test the Euclidean type bi-objective TSP. The DRL-MOA model is trained on 40-city instances and applied to approximate the PF of 40-, 70-, 100-, 150- and 200-city problems. For 100-, 150- and 200-city problems, we adopt the commonly used kroAB100, kroAB150 and kroAB200 instances [10]. The HV indicator and computing time are shown in TABLE III.

Fig. 9, 10 and 11 show the results for kroAB100, kroAB150 and kroAB200 instances. By increasing the number of iterations to 4000, NSGA-II, MOEA/D and our method can achieve a similar level of convergence for kroAB100 while MOEA/D performs slightly better. However, MOEA/D performs the worst in terms of diversity with all solutions crowded in a small region and its computing time is not acceptable.

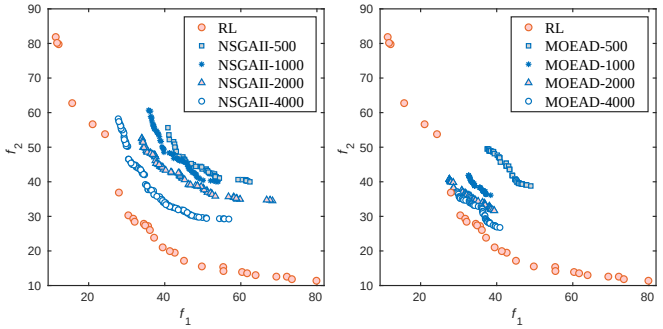
When the number of cities increases to 150 and 200, the PF obtained by DRL-MOA exhibits an enhanced performance in both convergence and diversity, as shown in Fig. 10 and 11. Even though 4000 iterations are conducted for NSGA-II and MOEA/D, there is still an obvious gap of performance between the two methods and the DRL-MOA.

In terms of the HV indicator as demonstrated in TABLE III, the DRL-MOA can always exhibit the best in comparison to MOEA/D and NSGA-II, even in the condition of 4000 iterations. Meanwhile, the computing time of using DRL-MOA is reasonable. Increasing the number of iterations for MOEA/D and NSGA-II can certainly improve the performance but would result in a large amount of computing time. It requires more than 150 seconds for MOEA/D to reach an acceptable level of convergence. The computing time of



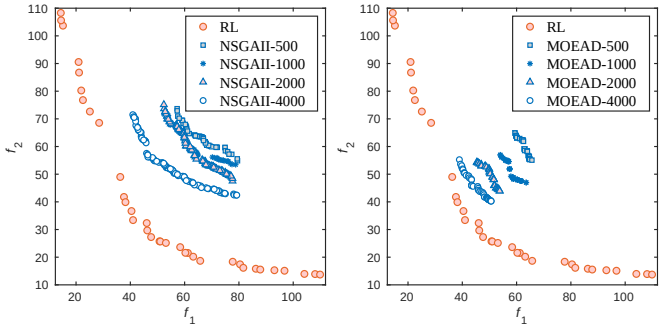
(a) DRL-MOA and NSGA-II (b) DRL-MOA and MOEA/D

Fig. 9. **KroAB100** Euclidean bi-objective TSP problem instance: the PF obtained using our method (trained using 40-city instances) in comparison with NSGA-II and MOEA/D. 500, 1000, 2000, 4000 iterations are applied respectively.



(a) DRL-MOA and NSGA-II (b) DRL-MOA and MOEA/D

Fig. 10. **KroAB150** Euclidean bi-objective TSP problem instance: the PF obtained using our method (trained using 40-city instances) in comparison with NSGA-II and MOEA/D. 500, 1000, 2000, 4000 iterations are applied respectively.



(a) DRL-MOA and NSGA-II (b) DRL-MOA and MOEA/D

Fig. 11. **KroAB200** Euclidean bi-objective TSP problem instance: the PF obtained using our method (trained using 40-city instances) in comparison with NSGA-II and MOEA/D. 500, 1000, 2000, 4000 iterations are applied respectively.

TABLE II
HV VALUES OBTAINED BY RL-MOA, NSGA-II AND MOEA/D. INSTANCES OF 40-, 70-, 100-, 150-, 200-CITY **MIXED** TYPE BI-OBJECTIVE TSP ARE TEST. THEIR COMPUTING TIME IS LISTED. THE BEST HV IS MARKED IN GRAY BACKGROUND AND THE LONGEST COMPUTING TIME IS MARKED BOLD

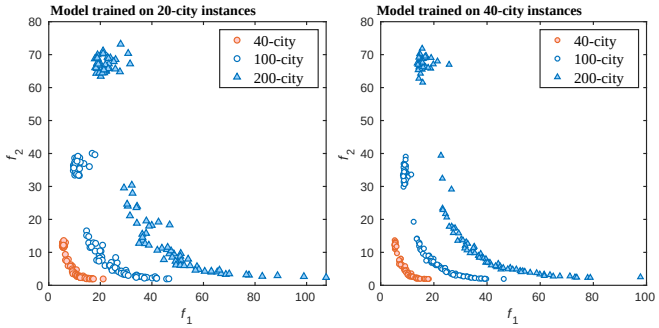
	40-city		70-city		100-city		150-city		200-city	
	HV	Time/s	HV	Time/s	HV	Time/s	HV	Time/s	HV	Time/s
NSGAII-500	1326	4.1	3904	4.2	7043	4.6	15180	5.7	25617	6.5
NSGAII-1000	1312	7.0	4102	9.6	7609	8.7	16058	12.6	28086	11.9
NSGAII-2000	1388	13.3	4267	16.8	8130	16.3	17445	21.4	29638	23.3
NSGAII-4000	1410	28.3	4367	32.7	8727	33.2	18033	40.5	30992	51.2
MOEA/D-500	1266	17.0	3879	17.7	7471	18.5	15606	20.5	26482	21.8
MOEA/D-1000	1314	34.5	4128	35.2	7688	35.9	16825	40.6	28395	41.9
MOEA/D-2000	1312	65.2	4194	68.5	8267	73.2	17691	79.4	30533	85.5
MOEA/D-4000	1281	**130.2**	4329	**136.0**	8442	**145.2**	18541	**157.6**	31834	**169.2**
RL-MOA	1398	2.7	4668	4.7	9647	6.6	22386	10.1	40354	12.9

TABLE III
HV VALUES OBTAINED BY RL-MOA, NSGA-II AND MOEA/D. INSTANCES OF 40-, 70-, 100-, 150-, 200-CITY **EUCLIDEAN** TYPE BI-OBJECTIVE TSP ARE TEST. THEIR COMPUTING TIME IS LISTED. THE BEST HV IS MARKED IN GRAY BACKGROUND AND THE LONGEST COMPUTING TIME IS MARKED BOLD

	40-city		70-city		100-city		150-city		200-city	
	HV	Time/s	HV	Time/s	HV	Time/s	HV	Time/s	HV	Time/s
NSGAII-500	1519	3.8	4464	4.1	8402	4.3	18172	5.2	31152	5.9
NSGAII-1000	1520	7.2	4545	7.7	9017	8.1	19900	9.6	32744	11.0
NSGAII-2000	1576	13.4	4739	15.7	9399	15.6	20015	20.1	34868	23.3
NSGAII-4000	1595	26.7	4965	29.5	9726	31.8	21689	40.2	36194	54.1
MOEA/D-500	1501	16.4	4543	17.1	8736	18.5	18637	20.3	33071	21.2
MOEA/D-1000	1469	33.6	4690	34.3	9361	36.5	20393	39.7	34556	42.4
MOEA/D-2000	1524	65.2	4669	69.5	9580	71.7	21117	78.6	36112	84.8
MOEA/D-4000	1512	**130.3**	4787	**135.2**	9720	**141.7**	21415	**156.9**	37606	**168.2**
RL-MOA	1603	2.6	5150	4.5	10773	6.3	24567	9.4	44110	12.9

NSGA-II is less, approximately 30 seconds, for running 4000 iterations. However, the performance for NSGA-II is always the worst amongst the comparing methods.

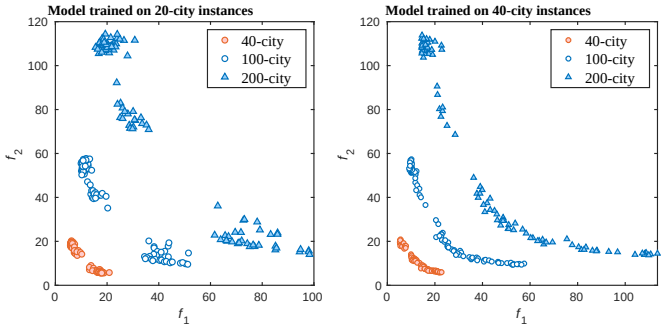
3) The impact of training on different number of cities: The forgoing models are trained on 40-city instances. In this part, we try to figure out whether there is a difference of training on 20-city instances. Euclidean instances and Mixed instances are both considered.



(a) Model trained on 20-city instances (b) Model trained on 40-city instances

Fig. 12. The two models trained respectively on 20- and 40-city **Mixed bi-objective TSP instances**. They are used to approximate the PF of 40-, 100-, 200-city problems.

As shown in Fig. 12 and 13, the number of non-dominated solutions obtained by training on 40-city instances are more than that obtained by training on 20-city instances. This condition is more serious for Euclidean instances, where a



(a) Model trained on 20-city instances (b) Model trained on 40-city instances

Fig. 13. The two models trained respectively on 20- and 40-city **Euclidean bi-objective TSP instances**. They are used to approximate the PF of 40-, 100-, 200-city problems.

significant number of solutions obtained by the 20-city model are crowded in several regions. The 20-city model exhibits a worse performance than the 40-city one. A possible reason is that, when training on 40-city instances, 40 city selecting decisions are made and evaluated in the process of training each instance, which are twice of that when training on 20-city instances. Loosely speaking, if both the two models use $120,000$ instances, the 40-city model are trained based on $120,000 \times 40$ cities which are twice of that of 20-city model. Therefore, the model trained on 40-city instances is better. And we can simply increase the number of training instances for 20-city model to improve the performance.

Lastly, it is also interesting to see that the solutions output by DRL-MOA are not all non-dominated. Moreover, these solutions are not distributed evenly (being along with the provided search directions). These issues deserve more studies in future.

4) Summary: Observed from the experimental results, we can conclude that the DRL-MOA is able to handle MOTSP both effectively and efficiently, Its advantages can be summarized as follows.

- Strong generalization ability. Once the model is trained on 40-city instances, it can be used to solve the MOTSP of any city number, e.g., 100-city or 200-city MOTSP.
- High level of convergence and wide spread of solutions. The performance of DRL-MOA is especially better for large-scale problems, such as 200-city MOTSP, than MOEA/D and NSGA-II.
- Reasonable computing time in comparison with the iteration-based evolutionary algorithms. Once the trained network model is available, it can be directly used to output the solutions by a simple feed-forward of the network.
- Modularity of the framework. It is easy to integrate any other solvers into the proposed DRL-MOA framework by just replacing the model of the subproblem. In addition to the TSP solver in this work, other solvers such as VRP [14] and Knapsack problem [30] can be integrated into the DRL-MOA framework to solve their multi-objective versions.

IV. Conclusion

Multi-objective optimization, appeared in various disciplines, is a fundamental mathematical problem. It has been a long time that evolutionary algorithms are recognized as suitable to handle such problem. However, evolutionary algorithms, as an iteration-based solver, are difficult to be used for on-line optimization. Moreover, without the use of a large number of iterations and/or a large population size, evolutionary algorithms do not scale well to large-scale optimization problems [11], [12], [10].

Inspired by the very recent work of Deep Reinforcement Learning (DRL) for single-objective optimization, this study, to the best of the authors knowledge, made the first attempt to apply DRL for multi-objective optimization, and has found very encouraging results. In specific, on the classic bi-objective TSPs, the proposed DRL-MOA exhibits significant better performance than NSGA-II and MOEA/D (two state-of-the-art MOEAs) in terms of the solution convergence, spread performance as well as the computing time, and thus, making a strong claim to use the DRL-MOA, a non-iterative solver, to deal with MOPs in future.

With respect to the future studies, first in the current DRL-MOA, a 1-D convolution layer which corresponds to the city information is used as inputs. Effectively, a distance matrix used as inputs can be further studied, i.e., using a 2-D convolution layer. Second, the distribution of the solutions obtained by the DRL-MOA are not as even as expected. Therefore, it is worth investigating how to improve the distribution of the obtained solutions. Lastly, in addition to bi-objective TSPs, other types of MOPs, e.g., continuous, and MOPs with more than two objectives can be further studied using the DRL method. Overall, multi-objective optimization by DRL is still in its infancy. It is expected that this study will be motivating more researchers to investigate this promising direction, developing more advanced methods in future.

References

[1] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[2] 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.

[3] L. Ke, Q. Zhang, and R. Battiti, "MOEA/D-ACO: A multiobjective evolutionary algorithm using decomposition and antcolony," *IEEE transactions on cybernetics*, vol. 43, no. 6, pp. 1845–1859, 2013.

[4] B. A. Beirigo and A. G. dos Santos, "Application of nsga-ii framework to the travel planning problem using real-world travel data," in *2016 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2016, pp. 746–753.

[5] W. Peng, Q. Zhang, and H. Li, "Comparison between MOEA/D and NSGA-II on the multi-objective travelling salesman problem," in *Multi-objective memetic algorithms*. Springer, 2009, pp. 309–324.

[6] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations research*, vol. 21, no. 2, pp. 498–516, 1973.

[7] D. Johnson, "Local search and the traveling salesman problem," in *Proceedings of 17th International Colloquium on Automata Languages and Programming, Lecture Notes in Computer Science,(Springer-Verlag, Berlin, 1990)*, 1990, pp. 443–460.

[8] E. Angel, E. Bampis, and L. Gourvès, "A dynasearch neighborhood for the bicriteria traveling salesman problem," in *Metaheuristics for Multiobjective Optimisation*. Springer, 2004, pp. 153–176.

[9] A. Jaszkiewicz, "On the performance of multiple-objective genetic local search on the 0/1 knapsack problem-a comparative experiment," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 402–412, 2002.

[10] T. Lust and J. Teghem, "The multiobjective traveling salesman problem: a survey and a new approach," in *Advances in Multi-Objective Nature Inspired Computing*. Springer, 2010, pp. 119–141.

[11] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "A Decision Variable Clustering-Based Evolutionary Algorithm for Large-scale Many-objective Optimization," *IEEE Transactions on Evolutionary Computation*, vol. In Press, 2016.

[12] M. Ming, R. Wang, and T. Zhang, "Evolutionary many-constraint optimization: An exploratory analysis," in *Evolutionary Multi-Criterion Optimization*, K. Deb, E. Goodman, C. A. Coello Coello, K. Klamroth, K. Miettinen, S. Mostaghim, and P. Reed, Eds. Cham: Springer International Publishing, 2019, pp. 165–176.

[13] D. H. Wolpert, W. G. Macready *et al.*, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.

[14] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," in *Advances in Neural Information Processing Systems*, 2018, pp. 9839–9849.

[15] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2692–2700.

[16] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

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

[18] 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*, 2016, pp. 1928–1937.

[19] K. Li, K. Deb, Q. Zhang, and S. Kwong, "An evolutionary many-objective optimization algorithm based on dominance and decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 694–716, 2015.

[20] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.
[21] K. Miettinen, *Nonlinear multiobjective optimization*. Springer Science & Business Media, 2012, vol. 12.
[22] R. Wang, Z. Zhou, H. Ishibuchi, T. Liao, and T. Zhang, "Localized weighted sum method for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 3–18, 2018.
[23] R. Wang, Q. Zhang, and T. Zhang, "Decomposition-based algorithms using pareto adaptive scalarizing methods," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 6, pp. 821–837, 2016.
[24] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.
[25] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.
[26] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in neural information processing systems*, 2000, pp. 1008–1014.
[27] G. Reinelt, "TSPLIBA traveling salesman problem library," *ORSA journal on computing*, vol. 3, no. 4, pp. 376–384, 1991.
[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[29] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.
[30] S. Gu and T. Hao, "A pointer network based deep learning algorithm for 0–1 knapsack problem," in *2018 Tenth International Conference on Advanced Computational Intelligence (ICACI)*. IEEE, 2018, pp. 473–477.

Kaiwen Li received the B.S., M.S. degrees from National University of Defense Technology (NUDT), Changsha, China, in 2016 and 2018.

He is a student with the College of Systems Engineering, NUDT. His research interests include prediction technique, multiobjective optimization, reinforcement learning, data mining, and optimization methods on Energy Internet.

Rui Wang received the B.S. degree from National University of Defense Technology (NUDT), Changsha, China, in 2008 and the Ph.D. degree from University of Sheffield, Sheffield, U.K., in 2013.

He is a Lecturer with the College of Systems Engineering, NUDT. His research interests include evolutionary computation, multiobjective optimization, machine learning, and various applications using evolutionary algorithms.

Tao Zhang received the B.S., M.S., Ph.D. degrees from National University of Defense Technology (NUDT), Changsha, China, in 1998, 2001, and 2004, respectively.

He is a Professor with the College of Systems Engineering, NUDT. His research interests include multicriteria decision making, optimal scheduling, data mining, and optimization methods on energy Internet network.