Multi-objective Pointer Network for Combinatorial Optimization

Le-yang Gao, Rui Wang, Chuang Liu, Zhao-hong Jia

Abstract—Multi-objective combinatorial optimization problems (MOCOPs), one type of complex optimization problems, widely exist in various real applications. Although meta-heuristics have been successfully applied to address MOCOPs, the calculation time is often much longer. Recently, a number of deep reinforcement learning (DRL) methods have been proposed to generate approximate optimal solutions to the combinatorial optimization problems. However, the existing studies on DRL have seldom focused on MOCOPs. This study proposes a single-model deep reinforcement learning framework, called multi-objective Pointer Network (MOPN), where the input structure of PN is effectively improved so that the single PN is capable of solving MOCOPs. In addition, two training strategies, based on representative model and transfer learning, respectively, are proposed to further enhance the performance of MOPN in different application scenarios. Moreover, compared to classical multi-objective meta-heuristics, MOPN only consumes much less time on forward propagation to obtain the Pareto front. Meanwhile, MOPN is insensitive to problem scale, meaning that a trained MOPN is able to address MOCOPs with different scales. To verify the performance of MOPN, extensive experiments are conducted on three multi-objective traveling salesman problems, in comparison with one state-of-the-art model DRL-MOA and three classical multi-objective meta-heuristics. Experimental results confirm that the proposed model outperforms all the comparative methods with only 20% to 40% training time of DRL-MOA.

Index Terms—Multi-objective combinatorial optimization, Deep reinforcement learning, Pointer network, Traveling salesman problem.

I. INTRODUCTION

MULTI-OBJECTIVE combinatorial optimization (MOCO) [1] is a class of mathematical optimization problems with discrete variables where multiple objectives are optimized simultaneously. MOCO problems (MOCOPs) can be formulated as follows [2]:

Minimize $F(X) = (f_1(X), f_2(X), \ldots, f_m(X))$

subject to $X \in D$

where $X = (x_1, x_2, \ldots, x_n)$ represents an $n$-dimensional discrete candidate solution, $F(X)$ and $D \in \mathbb{R}^n$ denote an $M$-dimensional objective space and an $n$-dimensional bounded discrete decision space, respectively. Due to several conflicting optimization objectives being considered simultaneously [3].

This work was supported in part by the National Natural Science Foundation of China (No. 71971002 and No. 71871076) and the National Science Fund for Outstanding Young Scholars (No. 62122093).

L. Gao and Z. Jia (Corresponding author) are with the School of Computer Science and Technology, and C. Liu is with the School of Internet, Anhui University, Hefei 230039, PR China (e-mail: e20101008@stu.ahu.edu.cn; zhjia@mail.ustc.edu.cn; chuang@mail.ustc.edu.cn).

Rui Wang is with the College of System Engineering, National University of Defense Technology, Changsha 410073, PR China (ruiwangnudt@gmail.com).

The most MOCOPs are NP-hard [4]. Evaluating the quality of solutions to MOCOPs involves Pareto dominance relationship. A set of Pareto optimal solutions, also known as the non-dominated solution set, is obtained by solving MOCOP.

In recent years, a large number of MOCOPs have been extensively studied. The multi-objective traveling salesman problem (TSP), one of the typical MOCOPs, can be used to model a broad range of real-life applications, such as vehicle routing [5], manufacturing [6], warehousing problem [7], computer disk defragmentation [8], etc. Given a list of cities in TSP, an optimum feasible route that visits each city once and returns to the starting city is required. The TSP is one of the classical NP-hard problems. Therefore, multi-objective TSP (MOTSP) can be used to evaluate the capabilities of multi-objective solvers effectively. It is important to emphasize that in the TSP mentioned in this paper the costs between any two cities are the same in both directions.

Due to being able to obtain multiple Pareto-approximated optimal solutions in one single simulation execution, multi-objective evolutionary algorithms (MOEAs) are widely used to solve MOCOPs [9], [10]. Furthermore, individual recombination based on genetic operators and few hyperparameters make MOEA able to balance exploitation and exploration effectively. NSGAII [11] and SPEA2 [12], two well-known MOEAs, have been widely employed in many real-world applications [13], [14]. To improve the diversity of solutions when solving multi-objective optimization problems (MOOPs), evolutionary algorithms (EAs) based on decomposition have been proposed, such as NSGAI1II [15] and MOEA/D [16], to maintain sub-populations that are uniformly distributed in objective space, in order to ensure that their individuals are as uniformly close to the Pareto front as possible.

Over the past few years, artificial intelligence technology has been increasingly applied to address combinatorial optimization problems (COPs). Inspired by sequence-to-sequence model, Vinyals et al. [17] proposed Pointer Network (PN). The PN with Attention mechanism [18] is trained to find approximate optimal solutions to TSP through supervised learning. However, high costs of constructing training samples with real optimal solutions hinder its application in large-scaled problems. Bello et al. [19] adopted deep reinforcement learning (DRL) to train PN, with objective functions as feedback signal. At the same time, the critic network [19] was employed as the baseline to improve the stability of training, which not only reduces the consumption of building training dataset, but also surpasses the original PN on addressing TSP and 0-1 knapsack problem. Nazari et al. [20] applied PN to vehicle routing problem (VRP) with dynamic properties.
and replaced the recurrent neural network (RNN) with a one-dimensional convolutional layer to receive the input. This model decreases the training time by 60% without weakening its optimization capability. At this point, the PN using DRL has demonstrated the following significant advantages over the other combinatorial optimization methods.

- Strong optimization capability, without requiring manually designed optimization strategies and heuristics.
- Fast optimization, that is, the trained model solves the problem through forward propagation without iterative process similar to EAs.
- Insensitive to problem scale, that is, the trained model can be applied to similar problems with different scales.

The use of DRL and PN to solve MOCOPs is still in the early stages. Li et al. [21] proposed a multi-objective deep reinforcement learning framework, called DRL-MOA, which divides the MOCOP into several sub-problems. These sub-problems are solved separately using the corresponding number of sub-PNs, which are trained collaboratively with a neighbourhood-based parameter transfer approach. According to the experimental results, DRL-MOA shows better optimization performance in three types of MOTSP problems. It is obvious that DRL-MOA requires training a large number of PNs to ensure that the complete Pareto front is obtained, which results in expensive training consumption. To further improve the utility of DRL models in the field of MOCO, this paper proposes a single-model multi-objective Pointer Network (MOPN) using DRL. The main contributions of this study are as follows:

- The input structure of the encoder of PN is improved with the incorporated objective weight, so that the single PN is also capable of solving MOCOPs, which reduces the training time of MOPN significantly compared to DRL-MOA.
- Two effective training strategies are proposed to further improve the training efficiency and optimization capability of the MOPN on MOCOPs.
- A valuable conclusion is obtained from experimental results. Instead of a larger-scaled training dataset, the training dataset with a suitable problem scale can make MOPN easier to obtain good training effectiveness and strong generalization ability.

The structure of the paper is organized as follows: Section II introduces the proposed MOPN in detail. The design of the comparative experiment is presented in Section III. The experimental results and discussions are described in Section IV. Finally, the conclusion of this study and future research directions are given in Section V.

II. Multi-Objective Pointer Network for Combinatorial Optimization

In this section, the proposed single-model MOPN to address MOCOP is introduced in detail. Firstly, the mathematical definition of MOTSP studied in this paper is formulated. Then, the framework of the proposed model is described, with an emphasis on the improvement of the input structure to the encoder. Finally, the training method and two proposed training strategies are presented.

A. Formulation of MOTSP

The MOTSP with M optimization objectives can be defined as follows: It is assumed that there is a set of n cities \( X \), denoted by \( X = \{x^1, x^2, \ldots, x^n\} \), where moving from city \( i \) to city \( j \) generates \( M \) kinds of costs, denoted by \((c_{ij}^1, c_{ij}^2, \ldots, c_{ij}^M)\). Tour \( Y \) that visits each city exactly once and returns to the starting point needs to be found while minimizing \( M \) cost functions simultaneously. The \( m \)-th cost function for tour \( Y \) can be formulated as follows:

\[
C_m(Y) = \sum_{i=1}^{n-1} c_{yi,yi+1}^m + c_{yn,1}^m, m = 1, 2, \ldots, M
\]

where \( y_i \) denotes the \( i \)-th visited city of \( Y \).

Suppose there is a set of weight vectors \( w^1, w^2, \ldots \) that uniformly distributed in the objective space. Take the bi-objective optimization problem as an example. A set of 100 vectors, \((1,0), (0.99, 0.01), \ldots, (0,1)\), is selected as weight vectors. Each weight vector \( w^K = \{w^K_1, w^K_2, \ldots, w^K_M\} \) represents a user preference for the MOCOP, and the elements in the weight vector represent the priorities of objectives, where larger value means higher priority. It is notable that \( w^K_1 + w^K_2 + \ldots + w^K_M = 1 \). Accordingly, the aggregated multi-objective cost function of tour \( Y \) with user preference \( w^K \) is formulated as follows:

\[
C(Y) = \sum_{m=1}^{M} w^K_m C_m(Y)
\]

B. Framework of MOPN

In this part, a generic model MOPN is designed for MOCOP, with a given set of inputs \( X = \{x^1, x^2, \ldots, x^n\} \). For MOTSP, each element in \( X \) represents the set of features of the corresponding city, i.e., \( x^i = \{x^i_1, x^i_2, \ldots, x^i_j\} \), where \( x^i_j \) represents the \( j \)-th feature of the \( i \)-th city. And the number of city features is determined according to the distribution of the problem. Moreover, the output of the proposed model is defined as \( Y = (y_1, y_2, \ldots, y_n) \), which can represent the selected tour in MOTSP, where \( y_i \) denotes the index of the city chosen to be visited at step \( t \). Accordingly, the probability of selecting output \( Y \) with input \( X \) through the probability chain rule is given as follows [22]:

\[
P(Y|X) = \prod_{i=1}^{n} P(y_{t+1}|y_t, y_2, \ldots, y_1, X_t)
\]

where \( X_t, \) representing the set of all available cities at step \( t \), is updated after each city is visited. \( P(y_{t+1}|y_t, y_2, \ldots, y_1, X_t) \) denotes the probability of selecting \( y_{t+1} \) as the next city to be visited after \( y_t \), based on the visited cities being \( y_1, y_2, \ldots, y_t \). The process of determining the next city to be visited according to conditional probabilities is referred to as a step. Therefore, the construction of a feasible tour can be obtained through performing this step iteratively. As a result, the process of mapping from input \( X \) to output \( Y \) can be modelled via PN.

In this paper, with the improved input structure of the encoder and the proposed training strategies, a novel PN model, called MOPN, is proposed to solve MOCOPs. Fig[1]
shows the structure of the MOPN. In the following, the three modules of the model, that is, the encoder, the decoder, and the attention layer, are described in detail, respectively.

1) Encoder:

The encoder can recognize the input sequence and store the features of cities in a vector with fixed size. Classical PN generally employs an RNN [26] as the encoder, which can aggregate both the input information and the input order, such as the words in a sentence and the order of the words. However, the inputs to many COPs are mutually independent entities with no temporal relationship, such as cities in TSP and VRP [23], items in knapsack problem [24], and jobs in scheduling problem [25], etc. For solving such problems, the RNN in encoder is no longer appropriate. Therefore, based on [20], an one-dimensional convolutional layer is employed, which not only maps the input to a high-dimensional vector, but also has a smaller computational complexity than RNN.

In addition, in order to satisfy the needs for solving MO-COPs, the building rules related to the input vector of the encoder are presented as follows:

a) The feature vector of city and weight vector of objectives are stitched together to form a input vector. The order of the objectives corresponding to the elements in the feature vector should be the same as that of weight vector.

b) In the feature vector, the feature dimension associated with each objective is kept consistent. If the feature dimension of one objective is less than others, it is necessary to pad the features according to the corresponding dimension. The padded features are set to 1.

By the building rules, the order and position of elements in the input vector are given practical meaning, which increase the amount of information that the input vector provides to the model. Meanwhile, the weights of objectives are also included into the input vector, which ensures that the information sent to all subsequent modules contains the objective weights. To explain the building rules of input vector specifically, an example for a two-objective MOTSP is provided. Suppose that the two-dimensional coordinates \((x_i, y_i)\) of one city, and the \(w_1\) and \(w_2\) denote the weights of the two objectives, respectively. Therefore, the input vector of the encoder for this problem can be illustrated in Fig[2].

As the feature of the second objective is only one-dimensional, according to the second rule, the fourth column in Fig[2] is added to be consistent with the feature dimensions of the other objectives. Furthermore, as can be seen from Fig[2] the objective weights are present at the end of input vector for each city.

Through the encoder, the feature vector of each city is transformed into a vector with length \(l\). These vectors, called the node embedding, are used as the input to the decoder. The node embedding of all cities can be merged into a matrix of \(n \times l\), called context embedding, which is a part of the input to the attention layer. It is worth noting that \(l\) represents the size of the one-dimensional convolutional layer of the encoder and \(n\) denotes the number of cities.

2) Decoder:

RNN is the main component of the decoder, which stores all the information of node embedding that has been input into the decoder. By combining this historical information with the node embedding of the last visited city, the decoder can output a decoding vector containing information about all the cities having been visited so far. This decoding vector, that is, the hidden layer state \(d_t\) of the RNN, is used as the input to the attention layer in step \(t\).

In order to minimize the consumption of training model, an RNN model of gated recurrent unit (GRU) [28], with similar performance of the LSTM [27] but with fewer parameters, is adopted in this study.

3) Attention layer:

In each step, the attention layer receives the context embedding \(c\) and the decoding vector \(d_t\) from the encoder and decoder, respectively, and then calculates the correlation of each city with the next visited city. The city with the highest correlation is most likely to be selected as the next city. The process is formulated as follows [20]:

\[
u^t_i = v^T_i \tanh(W_a[e_i; d_t]), \quad i = 1, 2, \ldots, n \tag{5}
\]
\[
a_t = \text{softmax}(u_t) \tag{6}
\]
\[
b_t = a_t c^T \tag{7}
\]
\[
\tilde{u}^t_i = v^T_i \tanh(W_b[e_i; b_t]), \quad i = 1, 2, \ldots, n \tag{8}
\]
\[
P(y_{t+1}|y_1, y_2, \ldots, y_t, X_t) = \text{softmax}(\tilde{u}_t) \tag{9}
\]
Algorithm 1 Forward propagation process of MOPN

**Input:** LIns \(X\), number of cities \(n\), zero vector \(s\) of length \(l\)

**Output:** tour \(Y\) with corresponding preference

1. Initialize a new empty tour \(Y\)
2. \(s ← \text{Encoder}(X)\)
3. for \(t ← 1, 2, \ldots, n\) do
4. if \(t = 1\) then
5. \(d_t ← \text{Decoder}(s)\)
6. else
7. \(d_t ← \text{Decoder}(e_i)\)
8. end if
9. \(\hat{u}_t ← \text{Attention layer}(e, d_t)\)
10. \(P ← \text{softmax}(\hat{u}_t)\)
11. Greedily select city \(i\) with the highest probability in \(P\)
12. \(Y ← Y \cup \{i\}\)
13. end for

where ";;" means the concatenation of two vectors, \(v_a\), \(v_c\), \(W_a\), and \(W_c\) are all learnable parameters, \(a_t\) and \(b_t\) denote the "attention" mask over the inputs and the context vector in step \(t\), respectively, and \(\hat{u}_t\) represents the likelihood that city \(i\) is chosen as the city to be visited in the next step. The softmax operator is used to normalize \(u_t\) and \(\hat{u}_t\). Ultimately, the probability of each city being visited at step \(t + 1\) is obtained.

Assume that the matrix recording the objective features of all cities is a root instance (RIns) of MOTSP. By a set of objective vectors uniformly distributed in objective space, a root instance can be extended into many leaf instances (LIns) with different preferences. Each LIns is denoted by a matrix similar to Fig.2. Based on the three main modules of MOPN, as shown in Fig.1, the forward propagation process is described in Algorithm 1.

During training, instead of greedily selecting, the MOPN selects the next city through sampling from the probability distribution.

After Algorithm 1 is executed with a LIns set of one RIns as the input, a set of solutions converging with different objective weights is obtained. The Pareto non-dominated solutions among the obtained solutions are selected to form the Pareto front of this RIns.

C. Training method for MOPN

To train the MOPN, the well-known Actor-Critic policy gradient method is adopted [29]. The policy gradient method can be used to iteratively train all trainable parameters of the encoder, decoder and attention layer based on the gradient estimate of the expected reward.

In this paper, the policy gradient method consists of an actor network and a critic network, whose parameters are \(\theta\) and \(\delta\), respectively. In brief, the main part of the actor network, the proposed MOPN, can map the given LIns of MOCOP to a probability distribution of solutions. Then, the actor network obtains an expected solution by sampling from this probability distribution and computes its objective function \(C\) as the reward according to Eq.(3). The critic network estimates the expected reward \(Z(X; \delta)\) of the solution obtained by the actor network based on information about the given LIns. In this paper, the proposed MOPN is trained using the policy gradient method based on batch training. Therefore, the average gradients of the actor and the critic networks for \(N\) LIns are formulated as follows [30]:

\[
d\theta = \frac{1}{N} \sum_{k=1}^{N} (C^k - Z(X^k; \delta)) \nabla_{\theta} \log P(Y^k|X^k) \tag{10}
\]

\[
d\delta = \frac{1}{N} \sum_{k=1}^{N} \nabla_{\delta} (C^k - Z(X^k; \delta))^2 \tag{11}
\]

where \(C\) and \(Z(X^k; \delta)\) represent the expected rewards obtained through the actor and the critic networks on LIns \(k\), respectively.

In addition, the policy gradient is an unsupervised training method, meaning that the optimal objective values of LIns is not required for training. Since the training dataset only consists of random LIns, the generation cost of dataset is almost negligible.

D. Proposed training strategy

Considering the high complexity of MOCOPs and the low cost of dataset generation, the training dataset of MOPN needs to contain a large number of LIns to improve the performance, which also leads to high time consumption for training. In order to balance between model performance and training time, as well as being applied to different application scenarios, the following three training strategies are proposed.

1) General training strategy (GTS):

In this strategy, the model to solve MOTSP with a given scale is obtained by training on the dataset with the corresponding scale. The disadvantages of this strategy are mainly two aspects. On the one hand, the training of MOPN will be time-consuming when the problem scale is large. On the other hand, this strategy makes the MOPN no longer insensitive to problem scale.

2) Training strategy based on representative model (TS-RM):

In this strategy, a representative model (RM) is obtained by training on the dataset with an appropriate problem scale. RM is expected to perform well on problems with most scales. In order to find the RM that meets the above requirements, the problem scale of the training dataset needs to be determined through a process of parameter adjustment. This strategy reduces the training time of MOPN while ensuring that the obtained model is insensitive to the problem scale.

3) Training strategy based on transfer learning (TS-TL):

Considering that the transfer learning is applicable to training PN [21], a training strategy based on transfer learning (TS-TL) is proposed to enhance the performance of MOPN. In this strategy, RM is adopted as an initial model for transfer learning. After the initial model is trained on the dataset with other scale for a short period of time, it will obtain better performance for problems of the corresponding scale. This strategy is able to balance between the time consumption of
training and the capability of MOPN. However, the obtained model is sensitive to problem scale.

The details of the parameter adjustment about these strategies is specified in Section III.C.

III. EXPERIMENT SETUP

In order to evaluate the performance of the proposed MOPN comprehensively, all-round comparative experiments are designed involving MOTSPs with three types of objective combinations in this section. The comparative methods include a state-of-the-art multi-objective optimization model DRL-MOA and three classical multi-objective meta-heuristics, that is, MOEA/D, NSGAII and NSGAIII. All training of the models is conducted on a single GTX 2080Ti GPU to compare their training efficiency. And all models and meta-heuristics are run on the Intel Core i7-10875H CPU with 16GB of RAM. The proposed models are implemented in python and are publicly available on GitHub \(^2\) to reproduce the experimental results and facilitate further research. The DRL-MOA is also publicly available on GitHub \(^2\). And the comparative meta-heuristics are implemented by calling the Pymoo Library \(^3\).

A. Introduction to testing problem

In this paper, two objectives of MOTSP are adopted in testing problem.

One of the objectives, tour length, is most widely considered in MOTSPs. When calculating this objective function using Eq.\(^2\), the cost of travel between two cities is defined as their Euclidean distance. Minimizing this objective can effectively reduce the fuel cost of travel. The MOTSP instances with this objective contain the coordinates \((x_i, y_i)\) of each city.

The second objective is the sum of the altitude differences between cities in a tour. It is usually used to reflect the smoothness of a travel route. Minimizing this objective can reduce fuel costs and increase travel comfort simultaneously. This objective is also calculated according to Eq.\(^2\), where \(c_{ij}\) represents the absolute value of the altitude differences between two cities. The MOTSP instances with this objective contain the altitude \(h_i\) of each city.

In experiments, the testing problems involving one and two types of optimization objectives are encoded as T1 and T2, respectively. O2 and O3 represent the testing problems have two and three optimization objectives, respectively. The codes of the testing problems are shown in TABLE I.

| Problem Code | Obj1     | Obj2     | Obj3        |
|--------------|----------|----------|-------------|
| T1O2         | Tour Length | Tour Length |             |
| T2O2         | Tour Length | Altitude Difference | - |
| T2O3         | Tour Length | Tour Length | Altitude Difference |

The two objectives of problem T1O2 are identical, complex and in conflict with each other. This kind of problems is suitable for verifying the exploitation and exploration capabilities of the proposed model. The objectives of problem T2O2 are tour length and altitude difference, respectively. This type of problem demonstrates the ability of the proposed model to optimize the problem with different objectives. In order to evaluate the performance of the model for optimizing problem with more than two objectives, problem ‘T2O3’ is also proposed.

It is worth mentioning that problems T1O2 and T2O3, with two same objectives of tour lengths, are presented in TABLE I. In this type of RIns, each city has two coordinates, which exist in different coordinate systems, respectively. Furthermore, the tour lengths in the two coordinate systems are calculated independently.

Considering different problem scales, each type of the testing problems is extended into 40-, 70-, 100-, 150- and 200-city MOTSPs.

Based on three types of testing problems and five problem scales, fifteen levels of testing problems are generated. For example, T1O2S100 represents problem T1O2 with 100-city. The training dataset of each level contains 500000 LIns according to \(^2\). All LIns are generated according to uniform distribution within \([0,1]\). Their weight vectors are also normalized. Different from the training dataset, the testing dataset contains RIns, instead of LIns. The DRL model needs to obtain the Pareto front by combining RIns with a set of weight vectors, as described in Section II.B. Moreover, the meta-heuristics can get the Pareto front directly by RIns. Considering that the meta-heuristics require a large number of iterations and take long computation time to solve MOTSPs, the number of RIns in the testing dataset is set to 20 to control the time consumption of experiment.

Besides the generated testing datasets, the widely used RIns kroAB100, kroAB150 and kroAB200 from TSPLIB \(^3\), corresponding to the 100-, 150- and 200-city T1O2 problems respectively, are also adopted. Note that the city features of these RIns also need to be normalized.

B. Introduction to evaluation indicators

In this paper, four evaluation indicators are selected to evaluate the performance of the comparative methods.

1) Hypervolume:

The hypervolume \(^3\), denoted by HV, describes the size of the region dominated by the non-dominated solution set. The larger the HV value, the closer the non-dominated solution set is to the real Pareto front, indicating better convergence and diversity of this set. The calculation of the HV relies on the selection of the reference point. In this paper, the maximum of each objective in the output solution sets obtained by all comparative methods is dynamically selected as the corresponding component of the reference point.

2) Spacing:

Spacing \(^4\), denoted by SPC, measures the uniformity of the propagation of the set of non-dominated solutions. The calculation of SPC involves the extreme point, the Euclidean distance \(D_i\) between consecutive non-dominated solutions and its average value \(\overline{D}\). The extreme points represent the non-dominated solutions that are optimal in each objective direction, respectively, and are selected in the same way as

\(^1\)https://github.com/gaoly/MOPN
\(^2\)https://github.com/kevin031060/RL_TSP_4static
the reference point related to HV. The SPC for the bi-objective problem is formulated in Eq. (12).

$$SPC = \frac{D_f + D_l + \sum_{i=1}^{N-1} |D_i - D_l|}{D_f + D_l + (N-1)D_l}$$

(12)

where $D_f$ and $D_l$ represent the distances from the two extreme points to their nearest point of the non-dominated solution set, respectively. The smaller the value of SPC, the more uniform the distribution of the solution set in the objective space. To be consistent with the definition of SPC, the three-objective problems are partitioned into three bi-objective problems, whose objectives are defined as every two of the three objectives. The non-dominated solutions to the three-objective problem are projected into the coordinate systems corresponding to the three bi-objective problems, and are selected according to Pareto dominance relationship to obtain Pareto fronts in the three coordinate systems. The average SPC value of these Pareto fronts is proposed to evaluate the uniformity in three-objective problems.

3) Training time:

The training time denotes the time taken by the model from initialization to completion of training. Since the meta-heuristics do not require training, MOEA/D, NSGAII and NSGAIII are not involved in the comparison of this indicator.

4) Running time:

The running time represents not only the time taken by the model to obtain the Pareto front through forwarding propagation, but also the time spent by the meta-heuristic to complete its iterations. Both training time and running time reflect the practicality and usability of the comparative methods.

C. Setting of parameters

The parameters of the proposed MOPN and all comparative methods are determined according to their original paper or preliminary experiments.

1) Generation of reference vectors:

In this paper, the set of weight vectors, that is uniformly distributed in the objective space, is used in the execution of MOPN, DRL-MOA, MOEA/D and NSGAIII. These vectors, referred to as reference vectors, are generated by the simplex-lattice design method [31]. The number of reference vectors about the two-objective and three-objective problems are set to 100 and 105, respectively. And the generation parameters of the simplex-lattice design method for the two-objective and three-objective problems are set to $(100, 0)$ and $(13, 0)$, respectively.

2) Setting of network parameters for MOPN and DRL-MOA:

The network parameters of MOPN are set to be similar to those of the models in [21] that are shown in TABLE II.

$D_{input}$ and $D_{output}$ in TABLE II represent the input dimension and output dimension of the network, respectively. $D_{problem}$ denotes the input dimension of the problem. “1D-Conv” means the one-dimensional convolutional layer. “Kernel size” and "stride" are important parameters for the convolutional calculation. In the decoder, a one-layer GRU network with the hidden size of 128 is employed. As can be seen from TABLE III, the critic network consists of four one-dimensional convolutional layers connected in a sequence. The final output of critic network is a one-dimensional evaluation value.

All the learnable parameters of the proposed model are initialized through the Xavier initialization method [33]. The Adam optimiser [34] is used to optimize them, with learning rate of 0.0001 and batch size of 200.

The parameters of each sub-model for DRL-MOA are also based on [21]. It is worth noting that the epoch number for training each sub-model of DRL-MOA is set to 5. This means that the total number of epochs of DRL-MOA is five times the number of sub-models. The numbers of sub-models of DRL-MOA are set to 100 and 105 for the two- and three-objective problems, respectively. Therefore, the corresponding total numbers of epochs for training are 500 and 525, respectively.

3) Setting of parameters for the training strategies:

In Section II.D, three training strategies are proposed to balance between model performance and training time. In these strategies, the problem scales of training datasets for RM need to be determined by preliminary experiments.

The selected MOTSP in the preliminary experiments is T1O2, due to sufficient complexity and shorter training time. In the preliminary experiments, six proposed models are trained on the datasets with problem scales of 10, 20, 30, 40, 50 and 100, respectively. The epoch number of training is set to 100. In addition, the problem scales of testing datasets are set to 10, 100 and 200 to evaluate the model performance for problems with different scales. Each model obtains the sets of non-dominated solutions for all 20 RIns in each testing dataset so that the average of their HV values is calculated.

The three box plots of the results of the preliminary experiments are shown in Fig.3A. According to Fig.3A on the MOTSP with 10-city, models trained on 40-city show better performance than those trained on the dataset with the same scale as that of the MOTSP. As can be seen from Fig.3B and Fig.3C, the models trained on 10-city, 20-city and 50-city perform poorly on the testing datasets with large scales. Moreover, the models trained on 40-city show the best optimization capability and robustness for the problems in Fig.3B and Fig.3C. Note that none of the models trained on 10-city and 100-city obtain the desired performance in the corresponding testing datasets. It can be concluded that: 1) the LIns with smaller scale do not have complete knowledge to solve MOTSP; 2) Secondly, the LIns with larger scale may cause MOPN to fall into local optimum during training. Therefore, the results of the preliminary experiments demonstrate that the GTS is weaker than TS-RM overall. The subsequent experiments do not include the GTS in Section II.D. Finally, the dataset with problem scale of 40 is selected as the training dataset for RMs.

According to the proposed TS-TL, RM should be trained on the dataset with the corresponding problem scale to improve performance, called the transfer learning phase. In this study, the scale of the training dataset used for this phase is set to 100 to control time consumption. To ensure that RM converges as much as possible, the number of training epochs is set to 100.
TABLE II: Network parameters of the MOPN

| Module  | Type     | Parameter                                      |
|---------|----------|-----------------------------------------------|
| Encoder | 1D-Conv  | $D_{input} = D_{problem}$, $D_{output} = 128$, kernel size=1, stride=1 |
| Decoder | GRU      | $D_{input} = 128$, $D_{output} = 128$, hidden size=128, number of layer=1 |
| Attention | -      | No hyper parameters                          |

| Critic Network |
|----------------|
| First layer    | 1D-Conv  | $D_{input} = D_{problem}$, $D_{output} = 128$, kernel size=1, stride=1 |
| second layer   | 1D-Conv  | $D_{input} = 128$, $D_{output} = 20$, kernel size=1, stride=1 |
| third layer    | 1D-Conv  | $D_{input} = 20$, $D_{output} = 20$, kernel size=1, stride=1 |
| forth layer    | 1D-Conv  | $D_{input} = 20$, $D_{output} = 1$, kernel size=1, stride=1 |

Considering the high time-consumption training on 100-city, the number of training epochs in the transfer learning phase is set to 30.

4) Setting of parameters for the meta-heuristics:

The classical multi-objective meta-heuristics, MOEA/D, NSGAII and NSGAIII, are selected as the comparative algorithms. MOEA/D is involved in comparative experiments for all problems, and NSGAII and NSGAIII are engaged in comparative experiments for two- and three-objective problems, respectively. The parameters of each meta-heuristic are set as follow:

Genetic Operation: The genetic operations used in the comparative algorithms MOEA/D, NSGA-II and NSGA-III are simulated binary crossover and polynomial mutation. According to [15], the crossover probability and the distribution index of the crossover operation are set to $p_c = 1.0$ and $\eta_c = 30$. Meanwhile, the mutation probability and the distribution index of the mutation operation are set to $p_m = 1/n$ and $\eta_m = 20$, where $n$ denotes the number of iterations. Moreover, the neighbourhood size of MOEA/D is set to 20.

Population Size: The population size of the comparative algorithms is set consistent with the number of reference vectors to ensure that all algorithms can search the objective space as fully as possible.

Termination Condition: According to [21], the maximum number of iterations of each comparative algorithm is set to 4000.

IV. EXPERIMENTAL RESULTS AND DISCUSSIONS

This section contains three parts. Section IV.A shows the comparison and analysis of MOPN and DRL-MOA in terms of training time. Section IV.B discusses the experimental results of the MOPN with other comparative methods on different testing problems. The results obtained from the comparative experiment are summarised in Section IV.C.

A. Comparison of training time

TABLE III shows the training time for models MOPN and DRL-MOA on all training problems. The first column lists the codes for the testing problems according to TABLE I. The first row lists the training phases of MOPN and DRL-MOA, respectively, where MOPN-1 denotes the phase for training RM and MOPN-2 represents the phase for transfer learning. "Time/s", "Epochs" and "Sum" in the second row represent the average training time for each epoch, the number of training epochs and the total training time on each problem, respectively.

According to columns 2 and 5 in TABLE III the training time per epoch of the MOPN is almost proportional to the problem scale of the training dataset. Rows 3 to 5 show that the training time of MOPN on problem T1O2 is slightly longer than that on problem T2O2. The reason is that, as an optimization objective, tour length is more complicated than altitude difference on T2O2. In addition, problem T2O3 has one more optimization objective than the other problems. Hence the training time of MOPN on this problem is longer than that of others. From column 10, it can be observed that the DRL-MOA has the similar training time on different problems. Since DRL-MOA requires training 5 epochs for each sub-model, the total number of training epochs is over 500.

In MOPN, the training time using TS-RM for each problem is equal to the time shown in column 4 of TABLE III. The training time using TS-TL is equal to the sum of time shown in columns 4 and 7 of TABLE III. By comparing the total training time, it can be seen that the training costs of MOPN...
TABLE III: Comparative results using training time

| Problem       | MOPN-1 | MOPN-2 | DRL-MOA |
|---------------|--------|--------|---------|
|               | HV     | SPC    | Time/s  | HV     | SPC    | Time/s  | HV     | SPC    | Time/s  |
| T1O2S40       | 0.64   | 3.09   | 364.01  | 0.65   | 3.14   | 350.74  | 0.69   | 3.96   | 304.43  | 0.66   | 44.70   | 261.46  |
| T1O2S50       | 0.65   | 3.97   | 2086.98 | 0.64   | 9.34   | 2100.28 | 0.64   | 9.34   | 1940.97 | 0.69   | 10.29   | 1366.14 |
| T1O2S100      | 0.66   | 15.65  | 5058.46 | 0.66   | 15.31  | 5109.48 | 0.66   | 15.31  | 4678.31 | 0.71   | 16.20   | 3041.24 |
| T2O2S40       | 0.69   | 18.99  | 9288.68 | 0.69   | 19.02  | 8444.26 | 0.73   | 19.84  | 4887.04 | 0.89   | 87.09   | 4872.98 |
| T2O2S50       | 0.72   | 3.23   | 216.61  | 0.72   | 3.23   | 215.81  | 0.71   | 3.30   | 216.37  | 0.77   | 4.11    | 185.39  |
| T2O2S100      | 0.67   | 5.70   | 640.23  | 0.68   | 5.82   | 630.92  | 0.76   | 6.61   | 468.45  | 0.69   | 49.48   | 400.81  |
| T2O2S100      | 0.71   | 10.20  | 1470.03 | 0.71   | 10.23  | 1438.18 | 0.78   | 11.12  | 1021.98 | 0.76   | 65.73   | 922.71  |
| T2O2S150      | 0.68   | 15.70  | 3078.86 | 0.70   | 15.74  | 2965.72 | 0.81   | 16.82  | 1835.10 | 0.85   | 78.77   | 1741.43 |
| T2O2S200      | 0.66   | 20.43  | 5200.67 | 0.70   | 20.74  | 4961.84 | 0.83   | 21.59  | 2713.52 | 0.91   | 88.13   | 2696.16 |

TABLE IV: Comparative results on two-objective TSP

| Problem Code | HV       | SPC       | Time/s  | HV       | SPC       | Time/s  | HV       | SPC       | Time/s  |
|--------------|----------|-----------|---------|----------|-----------|---------|----------|-----------|---------|
| T1O2S40      | 0.64     | 3.09      | 364.01  | 0.65     | 3.14      | 350.74  | 0.69     | 3.96      | 304.43  |
| T1O2S50      | 0.65     | 3.97      | 2086.98 | 0.64     | 9.34      | 2100.28 | 0.64     | 9.34      | 1940.97 |
| T1O2S100     | 0.66     | 15.65     | 5058.46 | 0.66     | 15.31     | 5109.48 | 0.66     | 15.31     | 4678.31 |
| T2O2S40      | 0.69     | 18.99     | 9288.68 | 0.69     | 19.02     | 8444.26 | 0.73     | 19.84     | 4887.04 |
| T2O2S50      | 0.72     | 3.23      | 216.61  | 0.72     | 3.23      | 215.81  | 0.71     | 3.30      | 216.37  |
| T2O2S100     | 0.67     | 5.70      | 640.23  | 0.68     | 5.82      | 630.92  | 0.76     | 6.61      | 468.45  |
| T2O2S150     | 0.71     | 10.20     | 1470.03 | 0.71     | 10.23     | 1438.18 | 0.78     | 11.12     | 1021.98 |
| T2O2S200     | 0.66     | 20.43     | 5200.67 | 0.70     | 20.74     | 4961.84 | 0.83     | 21.59     | 2713.52 |

B. Comparison of optimization performance

The following three parts are including in this subsection. 1) The experimental results of the proposed model and the comparative methods on the two-objective problems are specifically analysed. 2) The performance of all methods on the three-objective problem are discussed. In addition, the experimental results associated with parts 1) and 2) are obtained on the randomly generated testing datasets. 3) The evaluation indicators obtained from all methods on the generic RIns of TSPLIB are compared.

1) Results on two-objective TSP:

TABLE IV gives the comparative results of MOPN-RM, MOPN-TL, DRL-MOA, NSGAII and MOEA/D on the testing datasets of two-objective TSP, where MOPN-RM and MOPN-TL represent the proposed models using TS-RM and TS-TL, respectively. The first column in TABLE IV lists the codes of testing datasets based on the MOTSP in TABLE I. For example, T1O2S40 denotes the dataset corresponding to the T1O2 problem with problem scale of 40. "HV", "SPC", and "Time/s" in row 2 represent the indicators HV, SPC, and Running time, respectively. Each DRL model was trained five times to generate five independent models. The average values of evaluation indicators obtained by these models are shown in TABLE IV. Similar to the above models, the comparative meta-heuristics are also executed five times on each RIns. In TABLE IV the best results of each evaluation indicator on each testing dataset are shown in boldface.

It can be seen from columns 2, 5, 8, 11 and 14 in TABLE IV that the MOPN-TL obtains the best HV values on all testing datasets except on 40-city. The model that obtains the best HV value on 40-city dataset is MOPN-RM. This means that TS-TL can further improve the performance of MOPN based on RM. Nevertheless, MOPN-RM also has a competitive optimization capability compared to the other methods. According to columns 2, 5 and 8 in TABLE IV the larger the problem scales of testing datasets, the more significant the advantage of the MOPN over DRL-MOA in terms of HV. This indicates that MOPN is more suitable for optimizing large-scaled MOCOPs than DRL-MOA. As a result, it can be concluded that the proposed model outperforms the other comparative methods in terms of optimization capability.

Based on columns 3, 6, 9, 12 and 15 in TABLE IV it can be seen that the values of SPC of the MOPNs is better than those of all other methods on all RIns, that is, the Pareto front obtained by the MOPN is more uniform than others.

According to columns 4, 7, 10, 13 and 16 in TABLE IV the running time of DRL models on all problems is much lower than those of the comparative meta-heuristics. Moreover, the difference in running time between these models is extremely small.

Figs 4 and 5 illustrate the distribution of non-dominated solutions to the RIns T1O2S100-1 and T2O2S150-1, which represent the first RIns of T1O2S100 and T2O2S150 testing datasets, respectively. Similar to the results in TABLE IV the non-dominated solutions of each type of models in the two figures are all obtained through aggregating the results of the five independent models.

The X- and Y-axes in Fig 4 represent the two optimization objectives of the T1O2S100 problem. The closer the point to the axes, the higher the quality of the corresponding non-dominated solution on the relevant objectives. It can be observed from Fig 4 that the red points representing the solutions obtained by MOPN-TL are closer to the axis overall than the black points of MOPN-RM. Meanwhile, the two ends of the red point set are more dispersed than the set of black points, indicating that the proposed TS-TL improves the performance of MOPN and causes the Pareto front to be stretched towards both axes. The results of MOPN-RM and MOPN-TL shown in Fig 4 are consistent with those in TABLE IV. Moreover, the blue point set corresponding to DRL-MOA performs even slightly better than that of MOPN-TL in terms of Y-axis, but slightly worse than that of DRL-MOA, respectively.
be seen from Fig.4 that the convergence and diversity of the comparative meta-heuristics are both poor.

The X- and Y-axes in Fig.5 represent the two optimization objectives of T2O2S150 problem. Similar to Fig.4, the set of red points representing the solutions obtained by MOPN-TL in Fig.5 shows better convergence compared to MOPN-RM. At the same time, the Pareto front obtained by DRL-MOA has a large gap in the Y-axis direction.

It is worth mentioning in Figs.4 and 5 that MOPN fails to find the better solutions at the ends of Pareto front than DRL-MOA. The reason is related to the running characteristic based on decomposition of DRL-MOA. When the objective vector of a sub-problem is parallel to the coordinate axes, it is equivalent for DRL-MOA to solve a single objective TSP using PN, which makes it easier to find the optimal value of the corresponding objective.

2) Results on three-objective TSP:

TABLE V presents the comparative results of MOPN-RM, MOPN-TL, DRL-MOA, NSGAIII and MOEA/D on the testing datasets of three-objective TSP. Similar to TABLE IV, MOPN-TL in TABLE V obtains better HV values on the testing datasets with problem scales 70, 100, 150, and 200, respectively. It can be observed from columns 2, 5 and 8 that MOPN outperforms DRL-MOA in addressing the three-objective TSP. According to the results in TABLES IV and V, it can be seen that the optimization performance of MOPN on three-objective TSPs is more outstanding than that on two-objective TSPs. Meanwhile, the optimization effect of MOPN becomes more obvious as the problem scale increases. It can be found from columns 2, 3, 5 and 6 in TABLE V that MOPN-TL obtains better HV values and worse SPC values than MOPN-RM on three-objective TSPs, that is, TS-TL makes the Pareto front obtained by MOPN more non-uniform while improving the convergence performance of MOPN. Moreover, from columns 2, 4, 7, 10, 13 and 16 in TABLE V, the running time of DRL models remains superior to that of the meta-heuristics on more complex three-objective problems.

Fig.6 shows the distribution of the non-dominated solutions to the RIns T2O3S200-1. The X-, Y-, and Z-axes represent the optimization objectives of the three-objective TSP, including two tour lengths and an altitude difference, respectively. According to Fig.6, the convergence of the surface formed by the set of red points representing the solutions to MOPN-TL is better than those of MOPN-RM, indicating that the MOPN using TS-TL demonstrates better performance on addressing the three-objective TSP. The pink point set of DRL-MOA only displays a Pareto front for the objectives corresponding to the X- and Y-axes, indicating that DRL-MOA performs worse on optimizing the objective in terms of the z-axis. The single pink point in the lower right corner in Fig.6 is the solution obtained by the initial model of DRL-MOA. Similar to the case in Fig.5, the Z-axis direction is relatively easy to be optimized. After the initial model falls into a local optimum in this direction, DRL-MOA is difficult to explore the other directions through transfer learning. From the two sets consisting of green points and purple points, respectively, corresponding to NSGAIII and MOEA/D in Fig.6, there is a larger gap between the performance of these two meta-heuristics and that of MOPN-RM and MOPN-TL.

3) Results on the generic RIns:

TABLE VI presents the comparative results of MOPN-RM, MOPN-TL, DRL-MOA, NSGAIII and MOEA/D on the generic RIns of T1O2. The first column in TABLE VI lists the codes of the three RIns, where the number at the end of each name represents the problem scale. The optimal HV values for the three RIns are obtained by MOPN-TL. The other results are also consistent with those in TABLE IV.

C. Summary of the results

From the above experimental results, it can be found that MOPN is able to solve MOTSP effectively. Different from the classical meta-heuristics, MOPN only needs to consume a little time on forward propagation to obtain a Pareto front of MOCOP. At the same time, MOPN has a stronger optimization capability than meta-heuristics.
TABLE V: Comparative results on three-objective TSP

| Problem Code | MOPN-RM | MOPN-TL | DRL-MOA | NSGAIII | MOEA/D |
|--------------|---------|---------|---------|---------|--------|
| HV           | SPC     | Time/s  | HV      | SPC     | Time/s  | HV | SPC | Time/s |
| T2O3S40      | 5399.91 | 0.69    | 3.25    | 5370.14 | 0.70    | 3.29 | 4173.56 | 0.95 | 4.14 | 3214.15 | 0.73 | 65.05 | 2699.60 | 0.82 | 392.17 |
| T2O3S70      | 25461.34| 0.69    | 5.84    | 25699.57| 0.69    | 5.87 | 17975.21| 0.94 | 6.88 | 12568.10| 0.76 | 70.82 | 10824.24| 0.86 | 382.20 |
| T2O3S100     | 60910.83| 0.71    | 9.26    | 62281.74| 0.72    | 9.24 | 38310.55| 0.98 | 10.36| 24766.55| 0.80 | 83.54 | 20777.49| 0.86 | 410.18 |
| T2O3S150     | 185864.41| 0.72   | 14.19   | 192395.76| 0.74    | 14.20| 102340.02| 0.99 | 15.47| 59225.99 | 0.85 | 96.91 | 47857.07 | 0.91 | 407.99 |
| T2O3S200     | 474814.51| 0.75   | 19.18   | 493633.24| 0.75    | 19.29| 269076.38| 1.00 | 20.73| 137045.39| 0.87 | 113.26| 108606.35| 0.91 | 422.88 |

TABLE VI: Comparative results on the generic RIns

| Problem Code | MOPN-RM | MOPN-TL | DRL-MOA | NSGAIII | MOEA/D |
|--------------|---------|---------|---------|---------|--------|
| HV           | SPC     | time    | HV      | SPC     | time    | HV | SPC | time |
| kroAB100     | 1879.16 | 0.65    | 8.59    | 1906.25 | 0.66    | 8.77 | 1743.73 | 0.69 | 10.66| 1209.98 | 0.73 | 59.88 | 1092.02 | 0.90 | 368.28 |
| kroAB150     | 4880.51 | 0.66    | 14.01   | 4921.94 | 0.70    | 14.14| 4438.93 | 0.72 | 16.52| 2787.70 | 0.83 | 81.94 | 2652.00 | 0.92 | 432.75 |
| kroAB200     | 8887.19 | 0.71    | 21.97   | 9013.42 | 0.67    | 20.69| 8124.72 | 0.73 | 21.38| 4494.26 | 0.88 | 98.57 | 4514.44 | 0.95 | 439.07 |

V. CONCLUSIONS

MOCOPs, a type of classical mathematical optimization problems, widely appear in various real applications. Meta-heuristics have been applied to solve such problems for a long time. However, the iteration characteristic of meta-heuristics leads to a lot of time consumption during solving large-scaled MOCOPs, which makes meta-heuristics hard to be applied to real-time optimization.

Inspired by the recent work of DRL for single-objective optimization, this paper proposes a single-model MOPN to solve MOCOPs, whose running time is much less than that of meta-heuristics. Moreover, the training time of MOPN is also reduced by 60% to 80% compared to the state-of-the-art model DRL-MOA. Furthermore, the performance of the proposed model is significantly better than the comparative methods. The experimental results demonstrate that DRL has great potential and broad application prospects in addressing combinatorial optimization.

There are two valuable directions to be investigated in future research. On the one hand, other novel network architectures can be attempted to enhance model performance. To solve MOCOPs, this paper proposes a feasible single-model framework, where the networks of encoder, decoder and attention layer can be replaced by more efficient network architectures, such as graph convolutional network and multi-head attention layer. On the other hand, the optimization algorithms with DRL-based local search strategy can be taken into consideration to solve MOCOPs. Considering that DRL models do not require manual design, this type of algorithms is also appropriate for complex combinatorial optimization problems.

REFERENCES

[1] M. Á. Domínguez-Ríos, F. Chicanco, and E. Alba, “Effective anytime algorithm for multiobjective combinatorial optimization problems,” Information Sciences, vol. 565, pp. 210–228, 2021.
[2] G. Kirlik and S. Sayın, “A new algorithm for generating all non-dominated solutions of multiobjective discrete optimization problems,” European Journal of Operational Research, vol. 232, no. 3, pp. 479–488, 2014.
[3] Q. Liu, X. Li, H. Liu, and Z. Guo, “Multi-objective metaheuristics for discrete optimization problems: A review of the state-of-the-art,” Applied Soft Computing, vol. 93, p. 106382, 2020.

Fig. 6: Plots of non-dominated solutions on Ins. T2O3S200-1

Compared to the state-of-the-art model DRL-MOA, MOPN uses only one model to obtain the Pareto front for MOCOPs. Moreover MOPN shows better convergence and diversity than DRL-MOA on all types of testing datasets. The Pareto front obtained by MOPN is more homogeneous and without gaps. Additionally, MOPN-RM and MOPN-TL, compared to, DRL-MOA reduce the training time by about 80% and 60%, respectively.

By comparing MOPN-RM and MOPN-TL with each other, it can be seen that MOPN with TS-RM or TS-TL shows different advantages. With good optimization performance, MOPN-RM takes the shortest training time among all the comparative methods. Furthermore, MOPN-RM is insensitive to problem scales, so that a trained MOPN-RM model can be applied to problems with different scales. Therefore, MOPN-RM can be employed in the case where training time and generality of models are focused on. MOPN-TL can be employed in the other applications to obtain better optimization results than MOPN-RM.

By comparing MOPN-RM and MOPN-TL with each other, it can be seen that MOPN with TS-RM or TS-TL shows different advantages. With good optimization performance, MOPN-RM takes the shortest training time among all the comparative methods. Furthermore, MOPN-RM is insensitive to problem scales, so that a trained MOPN-RM model can be applied to problems with different scales. Therefore, MOPN-RM can be employed in the case where training time and generality of models are focused on. MOPN-TL can be employed in the other applications to obtain better optimization results than MOPN-RM.

REFERENCES

[1] M. Á. Domínguez-Ríos, F. Chicanco, and E. Alba, “Effective anytime algorithm for multiobjective combinatorial optimization problems,” Information Sciences, vol. 565, pp. 210–228, 2021.
[2] G. Kirlik and S. Sayın, “A new algorithm for generating all non-dominated solutions of multiobjective discrete optimization problems,” European Journal of Operational Research, vol. 232, no. 3, pp. 479–488, 2014.
[3] Q. Liu, X. Li, H. Liu, and Z. Guo, “Multi-objective metaheuristics for discrete optimization problems: A review of the state-of-the-art,” Applied Soft Computing, vol. 93, p. 106382, 2020.
[4] X. Cai, M. Hu, D. Gong, Y. Guo, Y. Zhang, Z. Fan, and Y. Huang, “A decomposition-based coevolutionary multiobjective local search for combinatorial multiobjective optimization,” Swarm and Evolutionary Computation, vol. 49, pp. 178–193, 2019.

[5] O. Cheikhrouhou and I. Khough, “A comprehensive survey on the multiple traveling salesman problem: Applications, approaches and taxonomy,” Computer Science Review, vol. 40, p. 100369, 2021.

[6] F. Lokin, “Procedures for travelling salesman problems with additional constraints,” European Journal of Operational Research, vol. 3, no. 2, pp. 135–141, 1979.

[7] J. A. Chisman, “The clustered traveling salesman problem,” Computers & Operations Research, vol. 2, no. 2, pp. 115–119, 1975.

[8] T. Zhang, L. Ke, J. Li, J. Li, J. Huang, and Z. Li, “Metaheuristics for the tabu clustered traveling salesman problem,” Computers & Operations Research, vol. 89, pp. 1–12, 2018.

[9] I. Giakikiouzis, R. C. Purushouse, and P. J. Fleming, “An overview of population-based algorithms for multi-objective optimisation,” International Journal of Systems Science, vol. 46, no. 9, pp. 1572–1599, 2015.

[10] A. Zhou, B. Qu, H. Li, S. 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.

[11] 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.

[12] E. Zitzler, M. Laumanns, and L. Thiele, “SPEA2: Improving the strength pareto evolutionary algorithm,” vol. 103, 2001.

[13] B. A. Beirigo and A. G. dos Santos, “Application of nsga-ii framework to the travel planning problem using real-world travel data,” in Advances in 2016 IEEE Congress on Evolutionary Computation. IEEE, 2016, pp. 746–753.

[14] J. Li, W. Luo, J. Zeng, and J. Xia, “Application of spea2 algorithm in web services selection,” in Advances in 2010 IEEE Youth Conference on Information, Computing and Telecommunications. IEEE, 2010, pp. 387–390.

[15] K. Deb and H. Jain, “An evolutionary many-objective optimization algorithm using reference-point-based non-dominated sorting approach, part i: solving problems with box constraints,” IEEE Transactions on Evolutionary Computation, vol. 18, no. 4, pp. 577–601, 2013.

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

[17] O. Vinyals, M. Fortunato, and N. Jaitly, “Pointer networks,” in Advances in Neural Information Processing Systems, 2015.

[18] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in Advances in International Conference on Learning Representations, 2014.

[19] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, “Neural combinatorial optimization with reinforcement learning,” in Advances in International Conference on Learning Representations, 2016.

[20] M. Nazari, A. Oroojlooy, L. V. Snyder, and M. Takáč, “Reinforcement learning for solving the vehicle routing problem,” in Advances in Neural Information Processing Systems, 2018.

[21] 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.

[22] S. Gu, T. Hao, and H. Yao, “A pointer network based deep learning algorithm for unconstrained binary quadratic programming problem,” Neurocomputing, vol. 390, pp. 1–11, 2020.

[23] S. Zanjani and S. Huber, “Objectives and methods in multi-objective routing problems: a survey and classification scheme,” European Journal of Operational Research, vol. 290, no. 1, pp. 1–25, 2021.

[24] Q. Zhou, J. Hao, and Q. Wu, “A hybrid evolutionary search for the generalized quadratic multiple knapsack problem,” European Journal of Operational Research, vol. 296, no. 3, pp. 788–803, 2022.

[25] C. P. Tomazella and M. S. Nagano, “A comprehensive review of branch-and-bound algorithms: Guidelines and directions for further research on the flowshop scheduling problem,” Expert Systems with Applications, vol. 158, p. 113556, 2020.

[26] 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,” in Advances in Conference on Empirical Methods in Natural Language Processing, 2014.

[27] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.
Chuang Liu received his Ph.D. degree in Management Science and Engineering from University of Science and Technology of China, Hefei, Anhui, P.R. China, in 2021. His research interests include scheduling problems, routing problems, reinforcement learning, and evolutionary algorithms.

Zhao-hong Jia received her Ph.D. degree in Management Science and Technology in 2008 from University of Science and Technology of China. She is a Professor with the School of Computer Science and Technology, Anhui University, Hefei. Her research interests include service computing, intelligent computation and its applications, service recommendation.