Hybridization of Evolutionary Algorithm and Deep Reinforcement Learning for Multiobjective Orienteering Optimization

Wei Liu, Rui Wang, Senior Member, IEEE, Tao Zhang, Kaiwen Li, Wenhua Li, Hisao Ishibuchi, Fellow, IEEE, and Xiangke Liao

Abstract—Multiobjective orienteering problems (MO-OPs) are classical multiobjective routing problems and have received much attention in recent decades. This study seeks to solve MO-OPs through a problem-decomposition framework, that is, an MO-OP is decomposed into a multiobjective knapsack problem (MOKP) and a traveling salesman problem (TSP). The MOKP and TSP are then solved by a multiobjective evolutionary algorithm (MOEA) and a deep reinforcement learning (DRL) method, respectively. While the MOEA module is for selecting cities, the DRL module is for planning a Hamiltonian path for these cities. An iterative use of these two modules drives the population toward the Pareto front of MO-OPs. The effectiveness of the proposed method is compared against NSGA-II and NSGA-III on various types of MO-OP instances. Experimental results show that our method performs best on almost all the test instances and has shown strong generalization ability.

Index Terms—Decomposition, deep reinforcement learning (DRL), evolutionary algorithms (EAs), multiobjective optimization, orienteering problems (OPs), pointer networks (PNs).

I. INTRODUCTION

THE ORIENTEEERING problem (OP), one of the most classical combinatorial optimization problems (COPs), regularly arises in the real world. They are widely applied to many practical problems, including the parcels delivery [1], military reconnaissance [2], rescue after disasters [3], and facility inspection [4]. Typically, an OP is described as follows. While there are a number of cities with different locations and profits, a decision maker needs to select some cities and determine the shortest tour for the selected cities such that the total profit collected from the visited cities is maximized. Since the first introduction of OP by Golden et al. [5] in 1987, a number of algorithms have been proposed to deal with OPs [6], [7], [8]. In addition, several variants of OP [9], [10] have been introduced, such as time-dependent OPs (TDOPs), team OPs (TOPs), and OPs with time windows (OPTWs), and multiobjective OPs (MO-OPs). These problems have received increasing attention in recent years.

This study focuses on MO-OPs, in which each node has multiple profits (e.g., profits in culture and entertainment). The objectives of MO-OPs are maximizing the total profits in multiple dimensions and minimizing the total tour length. Since MO-OPs are NP-hard, exact methods become unaffordable when the problem scale is large. Alternatively, metaheuristics, in particular, multiobjective evolutionary algorithms (MOEAs) [11], [12], become the mainstream for dealing with MO-OPs. To name a few, Schilde et al. [13] proposed a hybridized path relinking procedures with a Pareto ant colony optimization algorithm and an extended variable neighborhood search method, respectively. Experimental results on both benchmark instances and real-world instances demonstrated their good performance. In addition, Martí et al. [14] combined the GRASP and path relinking; and Matl et al. [15] proposed a large neighborhood search (LNS) method. Martín-Moreno and Vega-Rodríguez [16] developed a multiobjective artificial bee colony (MOABC) algorithm; and Bossek et al. [17] integrated local search techniques into NSGA-II. Mei et al. [18] proposed a multiobjective memetic algorithm (MOMA) and a multiobjective ant colony system (MACS) algorithm for a time-dependent MO-OP.

Although MOEAs have been widely applied to solve MO-OPs and other multiobjective optimization problems (MOPs), they have shown certain limitations [19], [20]. First, MOEAs are iteration-based methods that have to evolve for a large number of generations to find the global optimal solutions, especially in large-scale instances. This results in a long running time. Second, recombination operators in MOEAs have to be designed carefully, which requires rich domain knowledge in both MOEAs and MOPs. Furthermore, such algorithms are
usually optimized for specific tasks, which means that a slight change of the problem may require to redesign the algorithm.

In recent years, deep reinforcement learning (DRL) has shown promising performance in COPs [21]. Bello et al. [22] used a reinforcement learning (RL) algorithm to train a pointer network (PN) [23] for solving traveling salesman problems (TSPs). Nazari et al. [24] simplified the encoder module of the PN and introduced dynamic information as a part of inputs, making the model suitable for vehicle routing problems (VRPs). Kool et al. [25] proposed an attention model (AM) to solve the routing problems, including OPs. Li et al. [26] proposed a DRL-based multiobjective optimization algorithm (DRL-MOA) framework to solve MOPs by DRL algorithms. Experimental results in multiobjective TSP (MOTSP) instances showed its competitiveness in terms of both model performance and running time. Lin et al. [27] proposed a single preference-conditioned model to directly generate approximate Pareto solutions. This model was a learnable network and was trained by an efficient multiobjective RL algorithm. Experimental results showed its effectiveness on the MOTSP, multiobjective VRP (MOVRP), and multiobjective knapsack problem (MOKP). Pan et al. [28] applied an efficient optimization algorithm based on DRL to solve the permutation flow-shop scheduling problem (PFSP). It was proven that the proposed DRL-based optimization algorithm could get better results than the existing heuristics. More DRL-based intelligent optimization algorithms for manufacturing scheduling can be found in [29].

The main advantage of solving COPs by DRL methods is that the DRL model can generate solutions immediately once trained since it is an end-to-end method. Also, the generalization ability of DRL methods is attractive. The trained model is applicable to a set of similar instances with different scales [26]. Despite these advantages, the training of a DRL model costs much in terms of both time and computing resources. Also, it faces difficulty to solve complex problems involving a large number of decision variables such as large-scale MO-OPs. Moreover, whereas the application of DRL to COPs has been examined in many studies, its performance is still poor on some problems.

Therefore, it is difficult for existing methods to effectively solve large-scale MO-OPs. For a complex problem, decomposing it into several subproblems and solving these subproblems collaboratively is usually a good idea. In this way, not only does the complexity of the original problem get reduced but also the algorithms designed for the subproblems are more targeted. However, how to reasonably decompose the original problem and design effective algorithms for the subproblems remains challenging.

Since an OP essentially can be decomposed into two subproblems, i.e., city selection and Hamiltonian path planning [9], it is possible to solve OPs more effectively through problem decomposition. In this study, we propose to decompose an MO-OP into two basic COPs, i.e., an MOKP and a TSP and propose a hybrid optimization framework, namely, MOEA-DRL, to tackle MO-OPs. In MOEA-DRL, we treat the city selection subproblem as an MOKP and solve it using an MOEA. We treat the Hamiltonian path planning as a TSP and solve it using a pretrained DRL model. Once the DRL model is trained, it can immediately generate a solution (i.e., city visit sequence) for the cities selected by the MOKP solver, and feed the solution back to the MOKP solver, helping it adjust the selection of cities. Updating solutions iteratively eventually solves the given MO-OP. The global search ability of MOEAs and the powerful generalization ability of DRL models make the MOEA-DRL framework effective and applicable to large-scale MO-OPs. Specifically, in the MOEA-DRL framework, NSGA-II and NSGA-III are used as the MOKP solvers (i.e., two variants of MOEA-DRL are implemented), and an improved PN is used as the TSP solver. The performance of MOEA-DRL is evaluated by comparing it with NSGA-II and NSGA-III. These two MOEAs are implemented for MO-OPs and applied under various specifications of the termination conditions (i.e., the number of generations).

It is worth mentioning that in the MOEA-DRL framework DRL is much more suitable to serve as the TSP solver than other algorithms, for example, evolutionary algorithms (EAs), commercial optimization solvers (e.g., Gurobi [30]) and local search methods (e.g., 2-opt method). This is because the TSP solver has to be called many times during evolution and these algorithms are all iteration-based. Different from them, as an end-to-end method, a DRL model can output optimal solutions immediately. In addition, it is clear that the optimization quality of TSP greatly impacts the final performance of MOEA-DRL on MO-OPs. While 2-opt is usually effective, it was found inferior to well-trained DRL models on TSPs, especially for large-scale instances [25], [26]. These are the reasons why we use a DRL model as the TSP solver in the MOEA-DRL framework. We have also tried to improve it and conduct adequate pretraining of the DRL model. The main contributions of this study are summarized as follows.

1) A hybrid optimization framework for dealing with MO-OPs, namely, MOEA-DRL, is proposed. Due to the modularity and simplicity of the MOEA-DRL framework, it is possible to use any suitable MOEA and DRL algorithms. The MOEA-DRL framework inherits the global search ability of MOEAs and the strong generalization ability of DRL and performs well on a set of MO-OP instances.

2) The conventional PN is improved. Except for static information in the model inputs, distance information is embedded as a dynamic embedding, helping with city decoding. The introduction of a dynamic embedding mechanism effectively speeds up the convergence during model training.

3) The performance of the proposed MOEA-DRL framework is evaluated through computational experiments on various MO-OP instances (i.e., biobjective instances and three-objective instances) with different problem sizes. The MOEA-DRL framework shows clear advantages with respect to optimization quality, especially for large-scale instances.

4) High generalization ability of the MOEA-DRL framework is demonstrated. When the DRL module is trained
using 100-city instances, the MOEA-DRL performs well on MO-OP instances ranging from 20 cities to 1000 cities.

The remainder of this study is organized as follows. Section II first formulates the mathematical model of MO-OPs and then illustrates the proposed MOEA-DRL framework, including MOEAs and a dynamic PN (DYPN). After that, the setting of computational experiments is described in Section III, and the experimental results are analyzed in Section IV. Section V concludes this study and identifies some future research directions.

II. PROPOSED MODEL

A. Problem Formulation

An MO-OP is formulated as the following $(K + 1)$-objective problem where a subset $S$ of the given city set $V$ is selected and a tour for the cities in $S$ is determined

\[
\begin{align*}
\text{Maximize} & \quad f(S) = (f_1(S), \ldots, f_K(S), -f_{K+1}(S)) \\
\text{subject to} & \quad i = 1 \in S \subseteq V \\
& \quad \text{TourLength}(S) \leq T_{\text{max}} \\
\text{where} & \quad f_k(S) = \sum_{i \in S} s^i, k = 1, 2, \ldots, K \quad (4) \\
& \quad f_{K+1}(S) = \text{TourLength}(S). \\
\end{align*}
\]

In this formulation, $s^i$ is the $k$th profit from the city $i$, TourLength($S$) is the tour length for the cities in $S$, and $T_{\text{max}}$ is the upper bound for the tour length. It should be noted that the calculation of TourLength($S$) needs a TSP optimizer whereas the total profits in (4) are easily calculated. Thus, the MO-OP in (1)–(5) can be viewed as a bilevel MOP. On the outer level, multiobjective city selection is performed. Then, the tour length is minimized for the selected cities in the inner level. In this study, we propose to handle the optimal city selection by an MOEA and handle the tour length minimization by a DRL algorithm.

B. General Framework of MOEA-DRL

Without the preference of decision makers, MOEAs are the most common approaches for MOPs, and they can be generally classified into three categories, i.e., Pareto dominance-based, decomposition-based, and indicator-based approaches. For Pareto dominance-based MOEAs, e.g., NSGA-II, solutions are evaluated based on the Pareto dominance relation as well as an additional diversity metric. Decomposition-based MOEAs decompose an MOP into a set of subproblems by weighted scalarizing methods and solve these subproblems in a collaborative manner. Representatives include cellular-based MOGA [31], MOEA/D, MOEA/DD [32], and NSGA-III [33]. Indicator-based MOEAs search for the best solution set by optimizing a performance indicator, such as hypervolume (HV) [34] and inverted generational distance (IGD) [35]. The proposed framework MOEA-DRL can use any of these MOEAs, specifically depending on the choice of the MOKP solver.

In the proposed hybrid optimization framework, MOEA-DRL, an MO-OP is decomposed into an MOKP and a TSP. The MOKP is solved by MOEAs, resulting in a set of selected cities. Each individual in the MOEA presents a plan of city selection. Based on the selected cities, a DRL model is used as the TSP solver to output a Hamiltonian path. The obtained tour length for each individual (i.e., each selection plan of cities) is fed back to the MOEA and used in the constraint condition in (3) and the $(K+1)$th objective in (5). Nondominated solutions in the final population are the optimization results obtained by the MOEA-DRL where each solution has the corresponding Hamiltonian path. The general framework of MOEA-DRL is presented in Algorithm 1. As described in lines 2 and 3, the model training and population initialization are required before the process of evolution.

Compared with pure MOEAs, the problem decomposition strategy can reduce the complexity of the problem effectively. Since the MOEA module in MOEA-DRL only focuses on city selection, it can be coded through binary variables, which is much simpler than the traditional permutation coding method for MO-OPs. The reduction of problem complexity makes MOEAs efficient and applicable to large-scale instances. Compared with pure DRL models, the training of the DRL model in the MOEA-DRL framework is much easier. This is because in this framework the DRL model is designed for TSP with much less complexity than OP. It only needs geographical information about the city locations. Thus, once trained, it is applicable to MO-OPs with any city profit distribution, tour length limitation, and multiple objectives. Since the training of the DRL model is time consuming, these advantages are vital for its application to practical problems. Moreover, any MOEA can be employed as the MOKP solver and any end-to-end method for TSP [25], [36] can be integrated as the TSP solver in the MOEA-DRL framework. This framework is applicable to various MO-OP variants by simply adding related constraints to MOKP and TSP solvers.

In the MOEA-DRL framework, we test NSGA-II and NSGA-III as MOKP solvers and introduce a DYPN as the TSP solver. These will be described in detail in the following sections.

Algorithm 1 General Framework of MOEA-DRL

Input: instance set $\mathcal{M}$, maximum number of generations $MaxGen$, population size $N$

1: Initialize $gen \leftarrow 1$
2: Train DRL model for TSP: $\text{DRLModel} \leftarrow \text{REINFORCE}($M$)$
3: $\text{Pop} \leftarrow \text{Initialization}(N)$
4: while $gen \leq MaxGen$ do
5: $\text{Route} \leftarrow \text{DRLModel}($Pop$)$
6: $\text{Pop} \leftarrow \text{MOEA}($Pop$, \text{Route}, N)$
7: $gen \leftarrow gen + 1$
8: end while
9: $\text{Route} \leftarrow \text{DRLModel}($Pop$)$
10: $\text{Arc} \leftarrow \text{UpdateArc}($Pop$, \text{Route}$)
C. MOEAs for MOKP

Since we need to select cities first, this subproblem is modeled as an MOKP. Among various approaches proposed for MOKPs in the literature, MOEAs showed promising performance [37]. NSGA-II and NSGA-III are well-known and frequently used MOEAs and are embedded into our framework as MOKP solvers. The main difference between them is the individual selection strategy. While the NSGA-II evaluates solutions based on the Pareto dominance relation among solutions and the congestion degree of solutions, the NSGA-III introduces widely distributed reference points to maintain population diversity. It is generally thought that the NSGA-III is more suitable for high-dimensional MOPs. Following the general coding method of MOKPs, the binary coding method is used in the MOEA-DRL framework.

Considering that a good initial population can accelerate the convergence and improve the solution quality of EAs, this study proposes a greedy heuristic for population initialization on MO-OP instances. For each individual, cities are selected step by step based on an indicator, namely, the profit density. Assuming that the last visited city is i, the profit density vector \( (I_1, I_2, \ldots, I_N) \) is defined as the ratio of the profit vector \( (e_1, e_2, \ldots, e_N) \) and the Euclidean distance vector \( (e_1, e_2, \ldots, e_N) \), where \( e_{ij} \) denotes the Euclidean distance between the cities i and j. It is formulated as follows:

\[
I_j = s_j/e_{ij}, \quad i = 1, \ldots, N; \quad j = 1, \ldots, N(i \neq j) \tag{6}
\]

where \( s_j \) is calculated as the average

\[
s_j = \frac{1}{K} \sum_{k=1}^{K} s_{jk}. \tag{7}
\]

The indicator \( I_j \) in (6) shows how worthy city j is when the last selected city is i. The higher the value of \( I_j \) for city j is, the more likely j is to be selected in that step. Since each city can be visited only once, all the visited cities should be excluded from the next city selection procedure. The next city is then sampled based on a probability vector \( P' = (P'_j)_{j \in U_i} \), which is the normalization form of \( I_{U_i} = (I_j)_{j \in U_i} \) as follows:

\[
P' = \text{softmax}(I_{U_i}), \quad i = 1, \ldots, N \tag{8}
\]

where \( U_i \) is the set of unvisited cities when the last visited city is i. It is worth mentioning that since such a greedy population initialization method is designed to get as many profits as possible, we extra introduce a random initialization strategy to generate 25% solutions for those MO-OP instances, one of whose objectives is minimizing the tour length. The random initialization strategy randomly selects no more than \( 3/N \) cities to be visited for each individual, and the detailed route is decided by the TSP solver. The introduction of the random initialization strategy can greatly increase the population diversity during evolution.

D. Dynamic Pointer Network for TSP

Since the TSP solver is called many times, its computational efficiency is necessary. DRL is suitable for this purpose due to its end-to-end model property. The high performance of DRL on TSPs has been repeatedly demonstrated in the literature. Nazari et al. [24] improved the PN [23] by introducing an attention mechanism and verified its effect on TSP and VRP. Furthermore, in this work, based on the model of Nazari, we propose a DYPN for TSP, by introducing dynamic information into the model.

Taking a TSP instance with N cities as an example, the input and output of DYPN are the city locations \( X = \{x_i| i = 1, \ldots, N\} \) and the city permutation \( \pi = (\pi_1, \ldots, \pi_N) \), respectively. Following the probability chain rule, the policy \( p(\pi|r) \) for giving a solution \( \pi \) on case r can be defined as (9), where \( \theta \) represents the parameters to be learned.

\[
p_\theta(\pi|r) = \prod_{i=1}^{N} p_\theta(\pi_i|r, \pi_0, \ldots, \pi_{i-1}). \tag{9}
\]

Here, \( \pi_0 \) represents the initial state, that is, no city has been visited. At each decoding step \( t = 1, \ldots, N \), city \( \pi_t \) is selected from available cities based on the probability \( p_\theta(\pi_t|r, \pi_0, \ldots, \pi_{t-1}) \). It does not matter whether the depot is selected first or not since the tour of all the selected cities is always a circle in a TSP. The decoding step of the depot makes no difference to the total tour length.

The DYPN follows the encoder–decoder architecture like many other DRL models. The encoder is used to map the inputs into a high-dimensional knowledge vector, and the decoder is used to decode that vector to the desired sequence. The structure of DYPN is shown in Fig. 1. While a static embedding and a dynamic embedding are included in the encoder, an RNN, an attention layer, a context embedding layer, and a log-probability layer constitute the decoder.

1) Encoder: The encoder is structured to map the input information into a knowledge vector. Following the design in [24], the one-dimensional (1-D) convolution layer is used as the encoder in this work. As the sequential information of the TSP inputs is meaningless, traditional RNN [22] encoders are unnecessary here. In each 1-D convolution layer, the number of in-channels is the dimension of inputs, and the number of out-channels is set as \( d_h \) (\( d_h = 128 \) in this study). There are two 1-D convolution layers being introduced in this study: one for static embedding, and the other for dynamic embedding. The parameters of each 1-D convolution layer are shared...
among all the static inputs or the dynamic inputs. The inputs are defined as \(X = \{x^i | i = 1, \ldots, N\}\), in which each input \(x^i\) is formed as a set of tuples \(\{x^i_1, d^i_1 \} | i = 0, \ldots, T\). \(x^i\) and \(d^i\) are the static feature and the dynamic feature at step \(t\), respectively.

In the static encoder, since the city location is a 2-D vector, the number of in-channels of the 1-D convolution layer is set to 2. Thus, the static encoder maps an \(N \times 2\) vector to an \(N \times d_b\) vector (suppose there are \(N\) cities). This is the static embedding, and it is formed as \(\vec{s} = [\vec{s}_i | i = 1, \ldots, N]\).

Considering that the distance information between cities is important in planning a path, we introduce a Euclidian distance vector \(E_t = (e_{t1}, e_{t2}, \ldots, e_{tN})\) \((e_{tj}\) denotes the Euclidian distance between the city \(j\) and the city \(t\), which is the city selected at decoding step \(t\) as the dynamic feature. Before the dynamic encoding, data normalization is applied to speed up the model learning, as formulated in (10), where \(E_t^{\text{max}}\) and \(E_t^{\text{min}}\) are, respectively, the maximum and minimum value among the elements of the vector \(E_t\).

\[
d_t = \left\{ \begin{array}{l}
E_t^{\text{max}} - e_{ti} \quad |i = 1, \ldots, N \\
E_t^{\text{min}} - e_{ti}
\end{array} \right.
\]  
(10)

Similar to the process of the static embedding, the dynamic feature \(d_t\) is mapped by a 1-D convolution layer from the size of \(N \times 1\) to \(N \times d_b\). The embedded dynamic vector in step \(t\) is formulated as \(d_t\). As described in Fig. 1, the dynamic embedding \(d_t\) is only used in the attention layer.

2) Decoder: The decoder is used to decode the embeddings into a city sequence. In this work, the decoder is composed of an RNN, an attention layer, a context embedding layer, and a log-probability layer, as shown in Fig. 1. The RNN summarizes the information of the visited cities; the attention layer measures how important each city is in a decoding step; the context embedding layer generates a weighted static embedding; and the log-probability layer is used to calculate a probability vector for city selection.

As illustrated in Fig. 1, the decoding process works as follows. In timestep \(t \in \{1, \ldots, T\}\), the RNN (a gated recurrent unit (GRU) [38] in this study) takes the static embedding \(\tilde{s}^{t-1}\) of the last decoded city and its last memory state \(h_{t-1}\) (if \(t > 1\)) as inputs, and it outputs a new memory state \(h_t\), as formulated in (11). Since no city has been selected before step \(t = 1\), \(\tilde{s}^{0}\) in (11) is initialized by the embedding of a zero vector with the size of \(2 \times 1\)

\[
h_t = f_{\text{GRU}}(\tilde{s}^{t-1}, h_{t-1}).
\]  
(11)

The attention layer then takes the memory state \(h_t\), static embedding \(\tilde{s}\) of all cities, and the dynamic embedding \(d_t\) at step \(t\) as inputs. An attention vector \(a_t\) is calculated as shown in (12) and (13). \(v_a\) and \(W_a\) in (12) are learnable parameters, and “;” represents the concatenation of two vectors.

\[
u_t = v_a^T \tanh(W_a[\tilde{s}; d_t; h_t])
\]  
(12)

\[
a_t = \text{softmax}(u_t).
\]  
(13)

After that, the context embedding layer is used to produce a weighted static embedding vector \(c_t\) as follows:

\[
c_t = a_t \tilde{s}.
\]  
(14)

Finally, the probability vector mentioned in (9) is calculated as follows:

\[
u_t = v_c^T \tanh(W_c[\tilde{s}; c_t])
\]  
(15)

\[
p_t(\pi_t | s, r; \pi_{t-1}) = \text{softmax}(u_t)
\]  
(16)

where \(v_c\) and \(W_c\) are learnable parameters.

Based on the probability vector, the city to be visited is selected by an inference method [36]. In this work, the sampling search strategy is used in the training stage to increase the exploration ability, and the greedy search strategy is used in the validation stage. While the sampling search strategy samples one city based on the probability vector \(p_t(\pi_t)\) in each step \(t\), the greedy search strategy directly selects the city with the highest probability. The selected cities are masked to avoid repetitive visits. The order of the selected cities in the \(N\) decoding steps creates a solution \((\pi_1, \pi_2, \ldots, \pi_N)\) of the MO-OP.

3) Training Method: In this study, the REINFORCE algorithm [39] is used as an RL method to train the DYPN model. There are two networks to be trained: 1) an actor network and 2) a critic network. While the actor provides policies for determining the next action, the critic evaluates the reward of the given policy. For a TSP instance \(r\), the actor and the critic are parameterized by \(\theta\) and \(\phi\), respectively. The reward of the policy \(\pi\) given by the actor is formulated as \(L(\pi)\), and the evaluated reward given by the critic is formulated as \(V(r; \phi)\).

The loss function is defined as (17), and its gradient is formed as (18)

\[
\mathcal{L}(\theta | r) = E_{p_0(\pi | r)}[L(\pi) - V(r; \phi)]
\]  
(17)

\[
\nabla \mathcal{L}(\theta | r) = E_{p_0(\pi | r)}[\nabla L(\pi) \cdot V(r; \phi)]
\]  
(18)

The actor network used here is DYPN, and the critic is constructed as a multilayer dense network. The parameter settings for the actor and the critic are listed in Table I. The training process is described in Algorithm 2. We first initialize the actor and critic networks with random parameters \(\theta\) and \(\phi\) in \([-1, 1]\), respectively. In each training epoch, \(M\) instances are drawn from a problem set \(M\). For each instance \(r_m\), the actor provides a solution \(\pi\). Then, the reward \(L_m(\pi)\) and the evaluated reward \(V(r_m; \phi)\) are calculated. After the rewards and the approximated rewards are calculated for all \(M\) instances, the parameters of the actor and the critic are updated. Such a parameter update process is iterated for \(N_{\text{epoch}}\) times.

### III. Experimental Setup

All experiments in this study are conducted on a single RTX 3060 GPU and an AMD 8-Core 5800H CPU with 32-GB memory. The code is written in Python 3.8. All competitor algorithms are implemented with the help of the open library Geatpy,\(^1\) which provides various EAs. This section illustrates the instance setup, hyperparameter settings, and the evaluation procedure.

\(^1\)https://github.com/geatpy-dev/geatpy
The three-objective instance has the two total profit objectives and the total tour length objective. The mixed type instance has two objectives—to maximize the total single-criterion profit and to minimize the total tour length constraint in new instances may lead to performance deterioration or failure of the pure DRL models. This means that any change in the city profits or the total tour length constraint. This means that any change in the city profits or the total tour length constraint in new instances may lead to performance deterioration or failure of the pure DRL models.

In this work, 1 280 000 instances of 100-city TSP are randomly generated. The total tour length constraint is set as 2, 3, 4, 6, 10, and 15 in the 20-, 50-, 100-, 500-, and 1000-city test instances, respectively. Similarly, the reference point for the three-objective instances is (0, 0, −2), (0, −3), (0, −4), (0, −6), (0, −10), and (0, −15) for each problem size. The reference point corresponding to the profits type test instances is set as (0, 0) for all instances.

| Train Set: | As a machine learning method, the DRL model DYPN needs to be trained in advance. It is worth noting that its training requires only the location information of cities since it is designed for the TSP module. This is the case in any type of MO-OP instance. Different from the MOEA-DRL framework, the training of pure DRL models designed for OPs or MO-OPs [25], [26] requires not only the location information but also the city profits and the total tour length constraint. This means that any change in the city profits or the total tour length constraint in new instances may lead to performance deterioration or failure of the pure DRL models. In this work, 1 280 000 instances of 100-city TSP are randomly generated as the training set. The locations are all generated in a unit square [0, 1] * [0, 1] with the random seed of 1234. | Test Set: | Biobjective OP instances (i.e., profits type instances and mixed type instances), as well as three-objective instances, are considered in this study. The profits type instance has two profits in each city [13], and the two objectives are defined by the sum profits of the selected cities. The mixed type instance has two objectives—to maximize the total single-criterion profit and to minimize the total tour length [40]. The three-objective instance has the two total profit objectives and the total tour length objective. |
| --- | --- | --- | --- |
| To create test instances, the location of each city, one or two profits at each city, and a total tour length constraint are needed. The generation of the city profits also obeys the uniform distribution over [0, 1]. With the random seed of 12 345, 20-, 50-, 100-, 200-, 500-, and 1000-city test instances are generated. The total tour length constraint is set as 2, 3, 4, 6, 10, and 15 in the 20-, 50-, 100-, 500-, and 1000-city test instances, respectively. Note that there is only a single test instance for each of these problem sizes. HV is used as a performance indicator in this work. The reference point for calculating HV values is set as (0, −2), (0, −3), (0, −4), (0, −6), (0, −10), and (0, −15) for mixed type test instances with 20, 50, 100, 200, 500, and 1000 cities, respectively. Similarly, the reference point for the three-objective instances is (0, 0, −2), (0, 0, −3), (0, 0, −4), (0, 0, −6), (0, 0, −10), and (0, 0, −15) for each problem size. The reference point corresponding to the profits type test instances is set as (0, 0) for all instances. | To create test instances, the location of each city, one or two profits at each city, and a total tour length constraint are needed. The generation of the city profits also obeys the uniform distribution over [0, 1]. With the random seed of 12 345, 20-, 50-, 100-, 200-, 500-, and 1000-city test instances are generated. The total tour length constraint is set as 2, 3, 4, 6, 10, and 15 in the 20-, 50-, 100-, 500-, and 1000-city test instances, respectively. Note that there is only a single test instance for each of these problem sizes. HV is used as a performance indicator in this work. The reference point for calculating HV values is set as (0, −2), (0, −3), (0, −4), (0, −6), (0, −10), and (0, −15) for mixed type test instances with 20, 50, 100, 200, 500, and 1000 cities, respectively. Similarly, the reference point for the three-objective instances is (0, 0, −2), (0, 0, −3), (0, 0, −4), (0, 0, −6), (0, 0, −10), and (0, 0, −15) for each problem size. The reference point corresponding to the profits type test instances is set as (0, 0) for all instances. |

Algorithm 2 REINFORCE Algorithm

**Input:** problem set $\mathcal{M}$, number of instances $M$ for each training epoch, number of training epochs $N_{epoch}$

1. Initialize the actor and the critic network respectively with random weights $\theta$ and $\phi$
2. for epoch $= 1 : N_{epoch}$ do
3. reset gradients: $d\theta \leftarrow 0$, $d\phi \leftarrow 0$
4. sample $M$ instances from the problem set $\mathcal{M}$
5. for instance $m = 1, \ldots, M$ do
6. for step $t = 1 : N_{step}$ do
7. $\pi_t \leftarrow p_0(r_m, \pi_0, \ldots, \pi_{t-1})$
8. end for
9. compute reward $L_m^\pi$ ($\pi$)
10. compute estimated reward $V(r_m; \phi)$
11. end for
12. $d\theta \leftarrow \frac{-1}{M} \sum_{m=1}^{M} \left( L_m^\pi - V(r_m; \phi) \right) \nabla_{\theta} \log p_0(\pi | r_m)$
13. $d\phi \leftarrow \frac{-1}{M} \sum_{m=1}^{M} \nabla_{\phi} \left( L_m^\pi - V(r_m; \phi) \right)^2$
14. Update $\theta$ using $d\theta$ and $\phi$ using $d\phi$
15. end for

**A. Instance Setup**

**Training Set:** As a machine learning method, the DRL model DYPN needs to be trained in advance. It is worth noting that its training requires only the location information of cities since it is designed for the TSP module. This is the case in any type of MO-OP instance. Different from the MOEA-DRL framework, the training of pure DRL models designed for OPs or MO-OPs [25], [26] requires not only the location information but also the city profits and the total tour length constraint. This means that any change in the city profits or the total tour length constraint in new instances may lead to performance deterioration or failure of the pure DRL models.

In this work, 1 280 000 instances of 100-city TSP are randomly generated as the training set. The locations are all generated in a unit square $[0, 1] \times [0, 1]$ with the random seed of 1234.

**Test Set:** Biobjective OP instances (i.e., profits type instances and mixed type instances), as well as three-objective instances, are considered in this study. The profits type instance has two profits in each city [13], and the two objectives are defined by the sum profits of the selected cities. The mixed type instance has two objectives—to maximize the total single-criterion profit and to minimize the total tour length [40]. The three-objective instance has the two total profit objectives and the total tour length objective.

**B. Hyperparameters**

The hyperparameters used in the actor and critic networks are listed in Table I. 1D-Conv means the 1-D convolution layer. $D_{input}$, $D_{output}$, and $D_{problem}$, respectively, mean the dimension of inputs, outputs, and the problem data. Specifically, $D_{problem} = 2$ for the static embedding and $D_{problem} = 1$ for the dynamic embedding in the encoders.

The parameters in the training and testing phases are listed in Table II. Both the actor and critic networks are trained by the Adam optimizer [41] with the learning rate of 0.0001 and the dropout rate of 0.1 for 10 epochs, each of which contains 1 280 000 training instances. The training batch size is set as 20-, 50-, 100-, 200-, 500-, and 1000-city test instances are generated. The total tour length constraint is set as 2, 3, 4, 6, 10, and 15 in the 20-, 50-, 100-, 500-, and 1000-city test instances, respectively. Similarly, the reference point for the three-objective instances is (0, 0, −2), (0, 0, −3), (0, 0, −4), (0, 0, −6), (0, 0, −10), and (0, 0, −15) for each problem size. The reference point corresponding to the profits type test instances is set as (0, 0) for all instances.

**TABLE I**

Parameter Settings of the DRL Model

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

**Critic Network**

| Module       | Type      | Parameters                             |
|--------------|-----------|----------------------------------------|
| First layer  | 1D-Conv   | $D_{input} = 2 * hidden size$, $D_{output} = 20$, $kernel size = 1$, $stride = 1$ |
| Second layer | 1D-Conv   | $D_{input} = 20$, $D_{output} = 20$, $kernel size = 1$, $stride = 1$ |
| Third layer  | 1D-Conv   | $D_{input} = 20$, $D_{output} = 1$, $kernel size = 1$, $stride = 1$ |

**TABLE II**

Parameter Settings of Training and Testing

| Training Stage | Testing Stage | Hyper-parameters | Value | Hyper-parameters | Value |
|----------------|---------------|------------------|-------|------------------|-------|
| Number of epochs | 10             | Number of instances | 12,800,000 | Max number of generations | 30     |
| Number of instances | 12,800,000 | Batch size | 64 | Probability of crossover | Default |
| Optimizer | Adam | Probability of mutation | Default | |
| Dropout rate | 0.1 | Learning rate | 0.0001 | | |
C. Evaluation Procedure

In the proposed MOEA-DRL framework, NSGA-II and NSGA-III are, respectively, examined as MO-KP solvers, and DYPN is used as the TSP solver. To evaluate the effectiveness of MOEA-DRL, NSGA-II and NSGA-III are also directly used to solve MO-OPs. The maximum number of generations is set as 500, 2000, 10 000, and 40 000 in each of these two MOEAs (i.e., four different termination conditions are examined for comparison). The open library Geatpy of python is used for the implementation of NSGA-II and NSGA-III. In NSGA-II and NSGA-III, genetic operators are used in the recommended settings in Geatpy. During the evaluation, 31 runs of each algorithm are conducted on each test instance. The nondominated solutions shown in the figures in the following parts are obtained by a single run with the median performance from those 31 runs.

Two different coding methods of compared MOEAs (designed for MO-OPs) are examined—a single-chromosome-based permutation coding method [42] and a double-chromosome coding method [17]. As the depot is always fixed, it is not included in chromosomes. The single-chromosome permutation coding method first permutes the other cities. Following this permutation, these cities are visited sequentially as long as the total tour length constraint is not violated. The double-chromosome coding method contains two chromosomes for city selection and city permutation, respectively. The first chromosome uses a binary coding mode and the second chromosome uses a permutation coding mode. A coding example of these two methods is shown in Fig. 2. The two coding examples in Fig. 2 generate the same tour [Depot, 1, 4, 3, 6, Depot]. As for the double-chromosome coding method, while the cities numbered 1, 3, 4, and 6 are selected according to the first chromosome, the second chromosome gives their relative order, that is, [1, 4, 3, 6].

IV. EXPERIMENTAL RESULTS AND DISCUSSION

A. Training of the DRL Module

As a deep network method, DYPN requires a large number of instances for training, and the training process is time consuming. Dynamic embedding is introduced in this study to help with learning and speed up the convergence. In this section, we compare DYPN with PN [24] during the training stage. Taking the models trained on 20-city TSP instances as examples, the number of training batches is set as 2000 and the batch size is 1024. Each of the 100 training batches takes 40.8 and 37.8 s on DYPN and PN, respectively. Costs of DYPN and PN during the training are shown in Fig. 3. The cost is defined by the tour length to be minimized. As seen in Fig. 3, while PN needs about 1000 training batches to converge to the cost of 5, DYPN needs about 500 training batches. In addition to the improvement in learning speed, the optimization ability of DYPN is also stronger than PN even when both of them have been trained for 2000 batches.

DYPN trained on 100-city TSP instances is used as the TSP solver in the MOEA-DRL framework for all testing instances from 20 cities to 1000 cities. When training it on 1 280 000 instances for 10 epochs (the training batch size is 64), the total training time of DYPN is about 18 h. It is worth mentioning that, along with the increase of the training instance size, the number of learnable parameters increases explosively. Limited by the computing resources, DYPN in this study is not trained on instances with larger sizes.

B. Effectiveness of Two Coding Methods

The single-chromosome permutation coding method and the double-chromosome coding method are two frequently used coding methods for MO-OPs. They are compared on biobjective OPs using NSGA-II and NSGA-III (thus called “S-NSGA-II,” “S-NSGA-II,” “D-NSGA-III,” and “D-NSGA-III,” respectively) in this section. Experiments are conducted on 20-, 50-, 100-, 200-, 500-, and 1000-city instances. The test instances include both profits type and mixed type instances, which are generated in the same manner as described previously in Section III-A. The population size and the maximum number of generations are set as 100 and 500, respectively, in both NSGA-II and NSGA-III. Figs. 4 and 5 show the obtained nondominated solution sets.

We can see from these figures that the two coding methods show similar performance on small-scale instances (i.e., instances with 20, 50, and 100 cities). However, as the number of cities increases, the single-chromosome-based permutation coding method gradually shows its advantage. In both the mixed type and the profits type instances with 200 cities, algorithms with the single-chromosome permutation coding method outperform those with the double-chromosome coding method. Moreover, since it is difficult for the double-chromosome coding method to generate feasible solutions that take into account the constraints of the problem, the single-chromosome coding method is more effective. The effectiveness of the two coding methods is demonstrated through these experiments.
during evolution and there is no repair operator being introduced, this coding method loses its efficacy in large-scale instances with 500 and 1000 cities. Different from the double-chromosome coding method, solutions generated by the single-chromosome permutation coding method are always feasible, which shows more advantages in profits type instances. This is because the single-chromosome permutation coding prefers to visit more cities under the total tour length constraint, resulting in the preference for longer tours with larger profits as shown in Fig. 4. Due to the higher coding complexity, algorithms with the double-chromosome coding method consume more time, as shown in Fig. 6. Moreover, the time consumption gap between these two coding methods becomes larger and larger as the problem size increases. Since better results are consistently obtained from the single-chromosome permutation coding method, it is applied to all the compared MOEAs in this study.

C. Results on Mixed Type Biobjective OP

In this section, the MOEA-DRL framework is tested on the mixed type biobjective OP, including the 20-, 50-, 100-, 200-, 500-, and 1000-city instances. The MOEA-DRL frameworks using NSGA-II and NSGA-III as MOKP solvers are called “MOEA-DRL(NSGA-II)” and “MOEA-DRL(NSGA-III),” respectively. NSGA-II and NSGA-III are also directly examined in solving MO-OPs under four different termination conditions, i.e., 500, 2000, 10 000, and 40 000 generations. The average HV values and running time are shown in Table III. Fig. 7 shows the obtained nondominated solution sets on the instances with 100, 200, 500, and 1000 cities. The obtained solution sets for the 20-city and 50-city instances are not presented in Fig. 7 since all the competitor algorithms work well on these small-scale instances.

As shown in Table III, all algorithms have similar performance on small-scale instances with 20 and 50 cities. In these instances, NSGA-III-40000 has slightly higher HV values than MOEA-DRL. However, with the increase of the problem size, MOEA-DRL gradually shows its advantage over NSGA-II and NSGA-III. Especially, NSGA-II and NSGA-III exhibit an obviously inferior performance compared with our framework on the 200-, 500-, and 1000-city instances. It is not likely that NSGA-II and NSGA-III can achieve comparable performance to MOEA-DRL by increasing the number of generations, see Fig. 7, since the progress from 10 000 generations to 40 000 generations is minor. Another advantage of the
MOEA-DRL is that its obtained nondominated solutions show more diversity, for which the proposed population initialization method helps a lot. As for the NSGA-II and NSGA-III, due to the coding method, they pay more attention to maximizing the total profit but focus less on minimizing the tour length.

The total number of examined solutions in the MOEA-DRL framework is much smaller than that of MOEAs. While the population size in all the competitor algorithms is 100, the maximum number of generations is only 30 in our framework and up to 40,000 in NSGA-II and NSGA-III. When these algorithms are compared under a similar running time (excluding the training time for the DYPN model), it is clear that the MOEA-DRL framework outperforms MOEAs. This result validates the effectiveness of the proposed problem decomposition strategy. By planning routes using the DRL methods, the mission of the MOEAs changes from MO-OPs to MOKPs. For an instance with $N$ cities, while the size of the solution space of an MO-OP is $(N-1)!$, the size of the solution space of an MOKP is only $2^{N-1}$, which is much smaller. The decomposition strategy thus greatly reduces the problem complexity and helps MOEAs search for the optimal solutions.

There is little difference between NSGA-II and NSGA-III on both evaluation quality and running time in Table III. It is worth noting that in MOEA-DRL the DYPN model has to be pretrained. However, this is acceptable since in most scenarios model training can be offline or conducted in advance. Once DYPN is trained, the MOEA-DRL can be much more effective than MOEAs on large-scale instances as shown in Fig. 7.

To specify the output solutions, we select the knee solutions [43] of the nondominated solutions obtained by MOEA-DRL(NSGA-II) on each test instance and visualize their tours as shown in Fig. 8. When there is no preference for the multiple objectives, the knee solution is generally thought the most representative one. For a solution set, the knee solution is defined as the one, whose position in the objective space has the farthest distance from the straight line, which is connected by the two points at the extreme ends in the objective space. For a maximum biobjective optimization problem, the knee solution among the nondominated solution set in the objective space is shown as Fig. 9.

**D. Results on Profits Type Biobjective OP**

In this section, we test our framework on the 20-, 50-, 100-, 200-, 500-, and 1000-city profits type biobjective OP instances. The average HV values and running time are shown in Table IV. Fig. 10 shows the obtained nondominated solution sets on the instances with 100, 200, 500, and 1000 cities. The solution sets on 20- and 50-city instances are not shown since all algorithms perform comparably.

It is clear that the number of obtained solutions in Fig. 10 on the profits type instances is much smaller than that on
the mixed type instances in Fig. 7. This is explained as follows. In the mixed type instances, the total profit and the total tour length are clearly conflicting. As a result, a large number of widely distributed no-dominated solutions are obtained in Fig. 7. In profits type instances, two profits of each city are randomly generated. Thus, the two profit objectives are not necessarily conflicting. Moreover, under the total tour length constraint, each algorithm tries to visit cities as many as possible. For these reasons, only a few nondominated solutions are obtained in Fig. 10 for profits type instances, and their distribution is not diversified. However, profits type biobjective OPs are still practically important [13].

As shown in Fig. 10, although the increase in the number of generations gradually improves the convergence of solutions in NSGA-II and NSGA-III, the performance of NSGA-II and NSGA-III is far weaker than that of our framework, especially in large-scale instances. HV-based comparison results in Table IV also confirm this performance difference. That is, in Table IV, MOEA-DRL(NSGA-II) and MOEA-DRL(NSGA-III) significantly outperform NSGA-II and NSGA-III. The performance gap becomes larger and larger along with the increase of the problem size. Under a similar computation time, our framework outperforms NSGA-II and NSGA-III on all problem instances. The knee solutions of the nondominated solution sets obtained by MOEA-DRL(NSGA-II) on each test instance are visualized in Fig. 11.

E. Results on Three-Objective OP

In this section, we test our framework on the three-objective OP instances with 100, 200, 500, and 1000 cities. The three objectives are to maximize the two types of total profits and to minimize the tour length. Since NSGA-II and NSGA-III obtained similar results in these experiments, this section only presents the results obtained by NSGA-II under 500, 2000, 10,000, and 40,000 generations. The average HV values are shown in Table V for each algorithm on each problem instance. Fig. 12 shows the obtained nondominated solution sets.

As can be seen that MOEA-DRL significantly outperforms NSGA-II on all the test instances. Moreover, the gap in the average HV values between MOEA-DRL and NSGA-II becomes larger and larger when the problem size increases.

F. Compare With State-of-the-Art Learning Methods

Since learning methods are up-to-date for solving optimization problems these years [44], [45], in this section,
two state-of-the-art learning methods are implemented for performance comparison, including a DRL-MOA [26] and a multiobjective neural EA based on decomposition and dominance (MONEADD) [46]. Different from MOEAs, both DRL-MOA and MONEADD are end-to-end methods, and the corresponding neural networks are trained by an RL method and an EA, respectively. While the DRL-MOA and MONEADD are originally proposed to solve MOTSPs, we implement them for solving MO-OPs. The training instances of both the DRL-MOA and MONEADD are randomly generated 100-city single-objective OP instances, and they are tested on the profits type biobjective OP instances with 20, 50, 100, 200, 500, and 1000 cities. The detailed algorithm description and parameter settings in experiments are described as follows.

**DRL-MOA:** By weighting every objective of an MOP, DRL-MOA decomposes the problem into a set of single-objective subproblems, each of which is modeled as a neural network. Model parameters of all these networks are optimized by
TABLE VI
AVERAGE HV VALUES AND RUNNING TIME OBTAINED BY MOEA-DRL, NSGA-II, MONEADD, AND DLR-MOA. THE TEST INSTANCES ARE THE PROFITS TYPE BIJECTIVE OP INSTANCES. THE BEST HV VALUES ARE MARKED IN GRAY BACKGROUND

| Examined Solutions | 20-city | 50-city | 100-city | 200-city | 500-city | 1000-city |
|--------------------|---------|---------|----------|----------|----------|-----------|
| MOEA-DRL/NSGA-II   | 3.0E+03 | 35.3    | 28.9     | 2.48     | 65.2     | 6.44       | 104.0     | 274.9       | 222.3     | 2194.74 | 619.3  | 8472.9 | 1256.9 |
| NSGA-II-500        | 5.0E+04 | 28.4    | 3.3      | 155.5    | 3.0      | 248.2     | 2.9       | 542.7       | 7.4       | 1370.9   | 9.6    | 2187.1 | 9.1        |
| MONEADD            | 1.0E+02 | 20.9    | 3.4      | 68.0     | 2.8      | 213.5     | 3.6       | 359.3       | 5.7       | 1364.0   | 10.8   | 3980.1 | 17.5      |
| DLR-MOA            | 1.0E+02 | 22.0    | 10.2     | 152.8    | 11.3     | 358.0     | 14.5      | 1280.0      | 23.4      | 5032.6   | 37.3   | 10820.4 | 34.5      |

Taking the profits type biobjective OP instances as test instances, the performance of MOEA-DRL/NSGA-II, DRL-MOA, and MONEADD is shown in Fig. 13. The performance of NSGA-II-500 is also shown together for reference. The average HV values and running time of non-dominated solutions obtained by the comparison algorithms are listed in Table VI. Since both MONEADD and DRL-MOA are end-to-end methods, the number of their examined solutions is 100, which is dependent on the number of subproblems after decomposition. Whereas both MONEADD and

an RL method, and a neighborhood-based parameter-transfer strategy is used to speed up the training process of networks. The encoder is a fully connected neural network, referring to [26], and maps the inputs into a high-dimensional vector. While the decoder used in [26] is a PN that is designed for TSPs, in this work, we use the state-of-the-art AM [25], which is designed for OPs, as the decoder in the DRL-MOA framework. All the structural parameters of the models follow the references. Each test instance is decomposed into 100 subproblems, which are then modeled as 100 networks. The first network [i.e., with the weight vector of (0,1)] is trained on 1 280 000 randomly generated OP instances for 5 epochs, and other networks are, respectively, trained on 128 000 randomly generated OP instances for 5 epochs, based on the pretrained parameters of the previous network. The REINFORCE algorithm is used here for training models. The total training time of the DRL-MOA is about 10 h.

MONEADD: Different from traditional DRL methods, which train networks by RL algorithms, MONEADD proposes to train deep neural networks by EAs. In MONEADD, the parameter vector of a network is set as an individual. A set of neural networks, which are first initialized with random parameters, is the population in the EA. Decomposition and domination strategies are used for accelerating convergence. Since the code of the MONEADD is not open source, we tried our best to reproduce this algorithm according to the model description and parameter settings. It is worth mentioning that the parallel evolution strategy used in [46] is not used here, since it can only accelerate the training process but does not affect the optimization accuracy. Moreover, since the MONEADD generates solutions through long and short-term memory (LSTM) networks, whose input length is the same as the output length, the existing models are unsuitable for OPs, whose output length is variable. Therefore, we use the AM designed for OPs as the neural networks to be trained in the MONEADD. All the structural parameters of the AM follow the default settings in [25], and all the genetic operators are implemented following the recommended way in [46]. However, since several parameters are not specified in [46], we set them by experience. The crossover possibility and the mutation possibility are set as 0.8 and 0.001, respectively. The training instance size is 1280, the batch size is 128, the population size is 100, and the maximum number of generations is set as 500. The total training time of the MONEADD is about 9 h.

Fig. 13. Nondominated solution sets obtained by MOEA-DRL/NSGA-II, NSGA-II-500, MONEADD, and DRL-MOA. The test instances are the profits type bijective OP instances with 20, 50, 100, 200, 500, and 1000 cities.
DRL-MOA can generate solutions in a minute, however, the effectiveness of their solutions is not satisfactory. MONEADD obtains even worse solutions than NSGA-II-500 most of the time, except in the 500-city and 1000-city instances. Whereas DRL-MOA shows more superiority over NSGA-II-500 along with the increase of the problem size, both MONEADD and DRL-MOA are always much inferior to MOEA-DRL(NSGA-II).

It is understandable for this result. The parameter size of the deep neural networks in this work is so large (i.e., more than 10 thousand) that it is hard for EAs to optimize the solutions (i.e., model parameters). It seems that only simple networks, whose parameter sizes are small, are suitable to be trained by EAs. Moreover, the genetic operators used in the [46] may be too simple for a complex problem, such as the MO-OPs. How to design genetic operators purposefully is always challenging. As for DRL-MOA, whereas it performs well on MOTSPs, it seems that a more advanced encoding and decoding mechanism, as well as the model structure, should be proposed for solving the MO-OPs.

Furthermore, compared with the proposed framework MOEA-DRL, there is another disadvantage of solving MO-OPs by MONEADD or DRL-MOA. That is, the network parameters are required to be retrained once the objectives of the problems are changed, which is costly. Different from that, since the DRL module in MOEA-DRL is trained for TSPs, MOEA-DRL can be directly applied in MO-OPs with any objectives.

G. Training on Different Number of Cities

In the previous sections, the DRL model trained on 100-city TSP instances is always used in the MOEA-DRL framework on test instances with 20–1000 cities. To further examine the generalization ability of MOEA-DRL, we discuss the effectiveness of the DRL models trained on TSP instances with 20, 50, and 100 cities, respectively. Test instances in this section are 20-, 50-, 100-, 200-, 500-, and 1000-city mixed type biobjective OP instances, which are generated in the same manner as in Section III-A. The population size is set as 100 and the termination condition is 20 generations in all these three implementations of MOEA-DRL(NSGA-II) [i.e., namely, MOEA-DRL(NSGA-II)-20, MOEA-DRL(NSGA-II)-50, and MOEA-DRL(NSGA-II)-100]. Fig. 14 shows the obtained nondominated solution sets.

Based on Fig. 14, when the test instance size increases to 200, the performance of the model trained on 20-city instances starts to deteriorate. It finds only a small number of non-dominated solutions with low quality on 500- and 1000-city instances. Similarly, the model trained on 50-city instances shows weakness when the instance size increases to 1000. Whereas the MOEA-DRL framework shows a strong generalization ability to unseen instances, a large gap in the problem size between training instances and testing instances leads to clear performance degradation.

H. Summary of the Results

The experimental results can be summarized as follows.

1) The introduction of dynamic information to DYPN improves both convergence speed and solution quality.
2) The single-chromosome-based permutation coding method is more effective (i.e., faster evolution speed and higher solution quality) for MO-OPs compared to the double-chromosome coding method.
3) For all the mixed type biobjective instances, the profits type biobjective instances, and the three-objective instances, the MOEA-DRL framework shows obvious superiority in solution quality, especially for large-scale instances.
4) The MOEA-DRL framework shows strong generalization ability and performs well on 200-, 500-, and 1000-city instances even when the DRL model is trained on 100-city instances.

V. CONCLUSION

In this study, we proposed a new idea for solving MO-OPs by problem decomposition. More specifically, we decomposed an MO-OP into an MOKP and a TSP and proposed a hybrid optimization framework MOEA-DRL which hybridizes an MOEA and a DRL method to solve the MOKP and TSP, respectively. The MOEA-DRL framework was evaluated through computational experiments on randomly generated mixed type biobjective instances, profits type biobjective instances, and three-objective instances. Experimental results showed that the MOEA-DRL framework greatly outperforms NSGA-II and NSGA-III, especially, for large-scale problem instances. Besides, MOEA-DRL also showed obvious superiority over two state-of-the-art learning methods—DRL-MOA and MONEADD. It was also shown that the MOEA-DRL framework has a high generalization ability. For example, it
worked well on 1000-city instances even when the DRL model was trained on 100-city instances. Whereas the training of a DRL model takes hours, the trained model can be directly used for all test instances with different objectives and different numbers of cities.

One future research direction is to improve the performance of the MOEA-DRL framework by using more advanced MOEA and DRL methods. Especially, its efficiency improvement is important with respect to both the DRL training and the MOEA search. It is also a promising research direction to implement multiobjective algorithms based on the proposed MOEA-DRL framework for various variants of MO-OPs in the future.

REFERENCES

[1] Z. Wang and J. B. Sheu, “Vehicle routing problem with drones,” Transp. Res. B: Methodol., vol. 122, pp. 350–364, Apr. 2019.
[2] E. Dadiemir, R. Battia, M. Köksalan, and D. T. Oztürk, “UAV routing for reconnaissance mission: A multi-objective orienteering problem with time-dependent prizes and multiple connections,” Comput. Oper. Res., vol. 145, Jun. 2022, Art. no. 105882.
[3] G. D. Cubber, H. Balta, D. Doroftei, and Y. Baudoin, “UAS deployment and data processing during the balkans flooding,” in Proc. 12th IEEE Int. Symp. Saf. Security Rescue Robot. (SSRR), 2014, pp. 1–4.
[4] J. Nikolie, M. Burri, J. Rehder, S. Leutenegger, C. Haerzerler, and R. Siegwart, “A UAV system for inspection of industrial facilities,” in Proc. IEEE Aeros. Conf., 2013, pp. 1–8.
[5] B. L. Golden, L. Levy, and R. Vohra, “The orienteering problem,” Nav. Res. Logist., vol. 34, no. 3, pp. 307–318, 1987.
[6] G. Laporte and S. Martello, “The selective travelling salesman problem,” Discrete Appl. Math., vol. 26, nos. 2–3, pp. 193–207, 1990.
[7] R. Ramesh, Y.-S. Yoon, and M. H. Karwan, “An optimal algorithm for the orienteering tour problem,” ORSA J. Comput., vol. 4, no. 2, pp. 155–165, 1992.
[8] A. C. Leifer and M. B. Rosenwein, “Strong linear programming relaxations for the orienteering problem,” Eur. J. Oper. Res., vol. 73, no. 3, pp. 517–523, 1994.
[9] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden, “The orienteering problem: A survey,” Eur. J. Oper. Res., vol. 209, no. 1, pp. 1–10, 2011.
[10] A. Gunawan, H. C. Lau, and P. Vansteenwegen, “Orienteering problem: A survey of recent variants, solution approaches and applications,” Eur. J. Oper. Res., vol. 255, no. 2, pp. 315–332, 2016.
[11] K. Deb, A. Pratap, S. Agrawal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” IEEE Trans. Evol. Comput., vol. 6, no. 2, pp. 182–197, Apr. 2002.
[12] Q. Zhang, and H. Li, “MOEA/D: A multiobjective evolutionary algorithm based on decomposition,” IEEE Trans. Evol. Comput., vol. 11, no. 6, pp. 712–731, Dec. 2007.
[13] M. Schilde, K. F. Doerner, R. F. Hartl, and G. Kiechle, “Metaheuristics for the bi-objective orienteering problem,” Swarm Intell., vol. 3, no. 3, pp. 179–201, 2009.
[14] R. Martí, V. Campos, M. G. C. Resende, and A. Duarte, “Multiobjective GRASP with path relinking,” Eur. J. Oper. Res., vol. 240, no. 1, pp. 71–97, 2015.
[15] P. Matl, P. C. Nolz, U. Ritzinger, M. Ruthmair, and F. Tricoire, “Bi-objective orienteering for personal activity scheduling,” Comput. Oper. Res., vol. 82, pp. 69–82, Jun. 2017.
[16] R. Martín-Moreno and M. A. Vega-Rodríguez, “Multi-objective artificial bee colony algorithm applied to the bi-objective orienteering problem,” Knowl. Based Syst., vol. 154, pp. 93–101, Aug. 2018.
[17] J. Bossak, C. Grimmer, S. Meisel, G. Rudolph, and H. Trautmann, “Local search effects in bi-objective orienteering,” in Proc. Genet. Evol. Comput. Conf., 2018, pp. 585–592.
[18] Y. Mei, F. D. Salim, and X. Li, “Efficient meta-heuristics for the multi-objective time-dependent orienteering problem,” Eur. J. Oper. Res., vol. 254, no. 2, pp. 443–457, 2016.
[19] X. Zhang, Y. Tian, R. Cheng, and J. Jin, “A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization,” IEEE Trans. Evol. Comput., vol. 22, no. 1, pp. 97–112, Feb. 2018.
[45] Z. Ding, L. Chen, D. Sun, and X. Zhang, “A multi-stage knowledge-guided evolutionary algorithm for large-scale sparse multi-objective optimization problems,” Swarm Evol. Comput., vol. 73, Aug. 2022, Art. no. 101119.

[46] Y. Shao et al., “Multi-objective neural evolutionary algorithm for combinatorial optimization problems,” IEEE Trans. Neural Netw. Learn. Syst., early access, Sep. 2, 2021, doi: 10.1109/TNNLS.2021.3105937.

Wei Liu received the B.S. degree from the National University of Defense Technology, Changsha, China, in 2020, where he is currently pursuing the M.S. degree in management science and engineering. His main research directions are deep reinforcement learning, evolution algorithms, and multiobjective optimization.

Rui Wang (Senior Member, IEEE) received the bachelor’s degree from the National University of Defense Technology, Changsha, China, in 2008, and the Doctoral degree from the University of Sheffield, Sheffield, U.K., in 2013. He is currently an Associate Professor with the National University of Defense Technology. His current research interests include multiobjective optimization, and the development of algorithms applicable in practice.

Dr. Wang received the Operational Research Society Ph.D. Prize at 2016, and the National Science Fund for Outstanding Young Scholars at 2021. He is also an Associate Editor of several top journals, such as SEC, ESWA, and IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION.

Kaiwen Li received the B.S. and M.S. degrees from the National University of Defense Technology, Changsha, China, in 2016 and 2018, respectively, where he is currently pursuing the Ph.D. degree in management science and engineering. His research interests include prediction technique, multiobjective optimization, reinforcement learning, data mining, and optimization methods on energy Internet.

Wenhua Li received the B.S. and M.S. degrees from the National University of Defense Technology, Changsha, China, in 2018 and 2020, respectively, where he is currently pursuing the Ph.D. degree in management science and engineering. His current research interests include multiobjective evolutionary algorithms, energy management in microgrids, and artificial intelligence.

Hisao Ishibuchi (Fellow, IEEE) received the B.S. and M.S. degrees from Kyoto University, Kyoto, Japan, in 1985 and 1987, respectively, and the Ph.D. degree from Osaka Prefecture University, Sakai, Japan, in 1992. He is a Chair Professor with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China. His research interest is evolutionary multiobjective optimization.

Dr. Ishibuchi is currently an IEEE CIS AdCom Member from 2021 to 2023, an IEEE CIS Distinguished Lecturer from 2021 to 2023, and the General Chair of IEEE WCCI 2024 in Yokohama, Japan.

Tao Zhang received the B.S., M.S., and Ph.D. degrees from the 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.

Xiangke Liao received the B.S. degree from Tsinghua University, Beijing, China, in 1985, and the M.S. degree from the National University of Defense Technology, Changsha, China, in 1988. He is currently a Full Professor and the Dean of the School of Computer, National University of Defense Technology. His current research interests include parallel and distributed computing, high-performance computer systems, operating systems, cloud computing, and networked embedded systems.