Supplemental File of "Pareto Improver: Learning Improvement Heuristics for Multi-objective Combinatorial Optimization"

Zhi Zheng, Shunyu Yao, Genghui Li, Linxi Han, and Zhenkun Wang, *Member, IEEE*

I. Training detail

A. Problem Defination

MOTSP: MOTSP is a representative MORP problem [1]–[4]. Let $\{C^i = c^i_{j,k}, i = 1, \dots m, j = 1, \dots, n, k = 1 \dots, n\}$ be the m $n \times n$ cost metrics of an m-objective n-node MOTSP, where $c^i_{j,k}$ denotes the cost between nodes k and j for the i-th objective. In general, the i-th objective is defined as:

$$f_i(\boldsymbol{x}) = \sum_{t=1}^{n-1} c^i_{x_t, x_{t+1}} + c^i_{x_n, x_1}, i = 1, \dots, m, \tag{1}$$

where the solution $\boldsymbol{x} = (x_1, x_2, \dots, x_\tau)$ is the permutation of all n nodes, i.e., $x_t \in \{1, 2, \dots, n\}$ is unique for $t = 1, 2, \dots, n$. The length of a MOTSP solution is the same as the number of nodes, i.e., $\tau = n$.

Without loss of generality and following commonly used settings of learning-based heuristics, we evaluate the algorithm's performance using Euclidean MOTSP instances and Mixed MOTSP instances [1]. For Euclidean MOTSP instances, the cost $c^i_{j,k}$ between node j and node k with respect to each objective i is the Euclidean distance between the two 2-dimensional coordinates, i.e., $d = 2$ for each objective. On the other hand, for Mixed MOTSP instances, the cost with respect to one of the objectives is based on the 1-dimensional Manhattan distance (i.e., each node has a 1-dimensional coordinate on this objective), while the cost for the other objectives is based on the 2-dimensional Euclidean distance. To ensure consistency, the 1-dimensional coordinate is padded with 0 to make $d = 2$ for every objective. It is important to note that the instances are randomly generated by uniformly sampling the coordinates of each node from the range $[0, 1]$. In line with [1], our experimental study employs 2-objective and 3-objective MOTSP instances with 20-, 40-, 70-, and 100-nodes for each type (i.e., Euclidean MOTSP and Mixed MOTSP).

MOCVRP: We also employ the 2-objective CVRP [5], which considers the fairness of all transport costs as well as the total cost. The 2-objective CVRP can be expressed as follows:

$$\begin{aligned} \text{minimize} \quad & f_1(\boldsymbol{x}) = \sum_{j=1}^{q} C(\boldsymbol{\rho}^j), \\ & f_2(\boldsymbol{x}) = \max_{j \in \{1, \dots, q\}} C(\boldsymbol{\rho}^j). \\ \text{subject to} \quad & 0 \le \delta_i \le D, \quad i = 1, \dots, n, \\ & \sum_{i \in \boldsymbol{\rho}^j} \delta_i \le D, \quad j = 1, \dots, q. \end{aligned} \tag{2}$$

where $\boldsymbol{x}$ is a solution representing the complete route of the vehicle and consists of q sub-routes $\{\boldsymbol{\rho}^1, \boldsymbol{\rho}^2, \dots, \boldsymbol{\rho}^q\}$. Each sub-route starts from the depot x_0 and backs to x_0. $C(\boldsymbol{\rho}^j) = \sum_{t=0}^{n_j - 1} c_{x_t, x_{t+1}} + c_{x_{n_j}, x_0}$ denotes the cost of the sub-route $\boldsymbol{\rho}^j$ and n_j represents the number of customer nodes in it. $n = \sum_{j=1}^{q} n_j$ is the total number of customer nodes; δ_i denotes the demand of node i; D denotes the capacity of the vehicle. It should be noted that each customer node can be allowed to visit only once in the complete route $\boldsymbol{x}$.

The MOCVRP instances are generated in the same way as in [6]. To better reflect the local structure, we inherit the node feature design from [7], i.e., $d = 7$ and the length of a MOCVRP solution $\tau \ge n$.

B. Model Structure

Our model *improver* inherits the encoder-decoder architecture [7]. Its input at the t-th step for $t \in \{0, 1, 2, \dots T_I - 1\}$ has two parts: the solution $\boldsymbol{x}_t \in \mathbb{R}^\tau$ and the weight vector $\boldsymbol{\lambda}_c \in \mathbb{R}^m$. The output of the improver is a pair-wise heuristic action a_t that modifies the solution $\boldsymbol{x}_t$ to the next solution $\boldsymbol{x}_{t+1}$. We use $\psi(\boldsymbol{x}_t) \in \mathbb{R}^{m \times \tau \times d}$ to represent the features of $\boldsymbol{x}_t$, where d represents the dimension of the features of each node (i.e., $\psi(x^i_t)$ for $i \in \{1, 2, \dots, \tau\}$) with respect to each objective.

Encoder: The weight vector $\boldsymbol{\lambda}_c$ is inputted to the network during the initial embedding stage. Additionally, in accordance with [8], we incorporate sinusoidal positional encoding **PE** to represent the order information of the solution. So, the initial embedding $\boldsymbol{H}^{(0)} \in \mathbb{R}^{m\times\tau\times d_h}$ in terms of $\boldsymbol{x}_t$ can be calculated as

$$\boldsymbol{H}^{(0)} = \boldsymbol{W}_\psi \psi(\boldsymbol{x}_t) + \boldsymbol{W}_\lambda \boldsymbol{\lambda}_c + \boldsymbol{b} + \mathbf{PE}, \tag{3}$$

where $d_h = 128$ in this work. We use $\boldsymbol{h}_{j,i}^{(0)} \in \mathbb{R}^{d_h}$ for $i \in \{1, 2, \ldots, \tau\}$ and $j \in \{1, 2, \ldots, m\}$ to denote the embedding with respect to the i-th variable and the j-th objective. Accordingly, $\boldsymbol{H}_j^{(0)} = [\boldsymbol{h}_{j1}^{(0)}, \boldsymbol{h}_{j2}^{(0)}, \ldots, \boldsymbol{h}_{j\tau}^{(0)}]$.

Similar to [7], [9], we use L transformer encoder blocks to generate initial embeddings for multiple objectives. Each transformer encoder block comprises two sub-layers: a self-attention sub-layer (ATT) and a feed-forward (FF) sub-layer. Both sub-layers are connected by skip-connections [10] and followed by batch normalization (BN) [11]. The l-th ATT sub-layer can be expressed as follows:

$$\hat{\boldsymbol{h}}_{j,i}^{(l)} = \mathbf{BN}^{(l)}(\boldsymbol{h}_{j,i}^{(l-1)} + \mathbf{ATT}_{j,i}^{(l)}(\boldsymbol{h}_{j,i}^{(l-1)}, \boldsymbol{H}_j^{(l-1)})), \tag{4}$$

where $\mathrm{ATT}(\cdot)$ represents the self-attention operation [8], and $l \in \{1, 2, \ldots, L\}$. L is set to 3 in this work. We utilize single-head self-attention as the multi-head version does not significantly improve performance [7]. It is important to note that the attention mechanism and the subsequent FF sub-layer are computed separately for each objective. The l-th FF sub-layer can be formulated as

$$\boldsymbol{h}_{j,i}^{(l)} = \mathbf{BN}^{(l)}(\hat{\boldsymbol{h}}_{j,i}^{(l)} + \mathbf{FF}^{(l)}(\hat{\boldsymbol{h}}_{j,i}^{(l)})), \tag{5}$$

where $\mathrm{FF}(\cdot)$ indicates a 2-layer fully connected network with ReLU activation [12].

After the L transformer encoder blocks, we integrate the final $\boldsymbol{H}^{(L)}$ and represent the entire graph embedding $\boldsymbol{h}_g$ as

$$\boldsymbol{H} = \sum_{j=1}^{m} \boldsymbol{H}_j^{(L)}, \quad \boldsymbol{h}_g = \max_{i=1}^{\tau} \boldsymbol{H}_i, \tag{6}$$

where $\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_\tau] \in \mathbb{R}^{\tau\times d_h}$ and $\boldsymbol{h}_g \in \mathbb{R}^{d_h}$.

Decoder. Each variable's embedding $\boldsymbol{h}_i$ for $i \in \{1, 2, \ldots, \tau\}$ is integrated with the entire graph embedding $\boldsymbol{h}_g$ through

$$\boldsymbol{h}_i^c = \boldsymbol{W}\boldsymbol{h}_i + \boldsymbol{W}_g\boldsymbol{h}_g \tag{7}$$

After that, we implement the compatibility layer [7], [13] for decoding to predict the heuristic operator, which selects nodes pair-wise:

$$Y_{i,j} = \begin{cases} C \cdot \tanh\left(\dfrac{(\boldsymbol{W}_q\boldsymbol{h}_i^c)^\intercal(\boldsymbol{W}_k\boldsymbol{h}_j^c)}{\sqrt{d_k}}\right) & i \neq j \\ -\infty & i = j \end{cases}, \tag{8}$$

$$\boldsymbol{P} = \mathrm{softmax}(\boldsymbol{Y}), \tag{9}$$

where $i, j \in \{1, 2, \ldots, \tau\}$. We limit the values in the compatibility matrix $\boldsymbol{Y} \in \mathbb{R}^{\tau\times\tau}$ within $[-C, C]$ [7], [14], [15]. C is set to 10 in this work. To prevent selecting meaningless heuristic operators, the diagonal elements are masked. For both the training and the inference stages, the heuristic operators are sampled from the probability matrix P.

C. Training time

TABLE I
THE EPOCH TRAINING TIME OF DIFFERENT MOTSPS AND MOCVRPS.

Problem		20	40	100
Euclidean MOTSP	2-objective	53m	91m	172m
	3-objective	62m	120m	203m
Problem		20	40	100
Mixed MOTSP	2-objective	53m	91m	172m
	3-objective	62m	120m	203m
Problem		20	50	100
MOCVRP	2-objective	76m	132m	235m

All the models are trained with 50 epochs on a Tesla-V100 GPU, and each epoch includes $5,000$ randomly generated instances divided into 10 batches. The training time for each epoch is listed in the Table I. It is noteworthy that the training time required for Mixed MOTSPs is consistently the same as that of Euclidean MOTSPs with identical instance sizes.

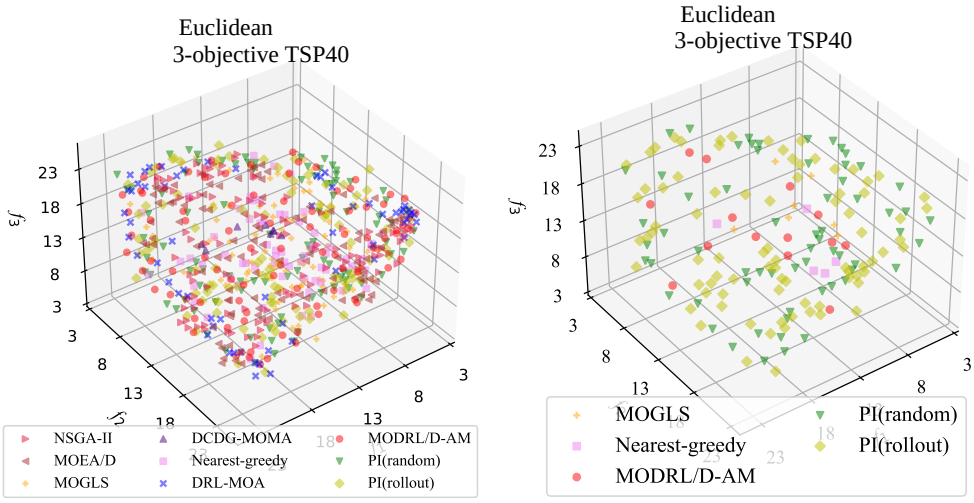
II. Experiment detail

Algorithm 1 Testing process

Input: The actor model $\pi_{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta}$; k uniformly distributed weight vectors $\{\boldsymbol{\lambda}^1, \boldsymbol{\lambda}^2, \ldots, \boldsymbol{\lambda}^k\}$; Archive $\mathcal{S}_p$; Initial population $\mathcal{S}$ with size p_{size}, total steps $T = 8000$, $T_I = 4$

Output: The improved population $\mathcal{S} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{p_{size}}\}$

for $i = 0, 1, \ldots, T-1$ **do**
  if $mod(i, 400) == 0$ **then**
    Add non-dominated solutions in $\mathcal{S}$ to $\mathcal{S}_p$
  end if
  Add $\arg\min_{\boldsymbol{x}\in\mathcal{S}} g(\boldsymbol{x}|\boldsymbol{\lambda}^i), i \in \{1, 2, \ldots, k\}$ to $\mathcal{S}_p$
  Random sample $\boldsymbol{\lambda} \in \{\boldsymbol{\lambda}^1, \boldsymbol{\lambda}^2, \ldots, \boldsymbol{\lambda}^k\}$
  $\boldsymbol{x}_0 \leftarrow \arg\min_{\boldsymbol{x}\in\mathcal{S}} g(\boldsymbol{x}|\boldsymbol{\lambda})$
  $\boldsymbol{x}^* \leftarrow \boldsymbol{x}_0$
  for $t = 0, 1, \ldots, T_I - 1$ **do**
    $a_t \sim \pi_{\boldsymbol{\theta}}(a_t|\boldsymbol{x}_t, \boldsymbol{\lambda})$
    $\boldsymbol{x}_{t+1} \leftarrow$ Transition$(\boldsymbol{x}_t, a_t)$
    $\boldsymbol{x}^*_{t+1} \leftarrow \arg\min(g(\boldsymbol{x}_{t+1}|\boldsymbol{\lambda}), g(\boldsymbol{x}^*_t|\boldsymbol{\lambda}))$
  end for
  if $\mathcal{S}_p = \mathcal{S}$ **then**
    $\boldsymbol{x}^r \leftarrow \boldsymbol{x}_0$
  else
    $\boldsymbol{x}^r \leftarrow \arg\max_{\boldsymbol{x}\in\mathcal{S}-\mathcal{S}_p} g(\boldsymbol{x}|\boldsymbol{\lambda})$.
  end if
  Replace the solution $\boldsymbol{x}^r$ by $\boldsymbol{x}^*_{\boldsymbol{T_I}}$ in Population $\mathcal{S}$
end for
return $\mathcal{S} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{p_{size}}\}$



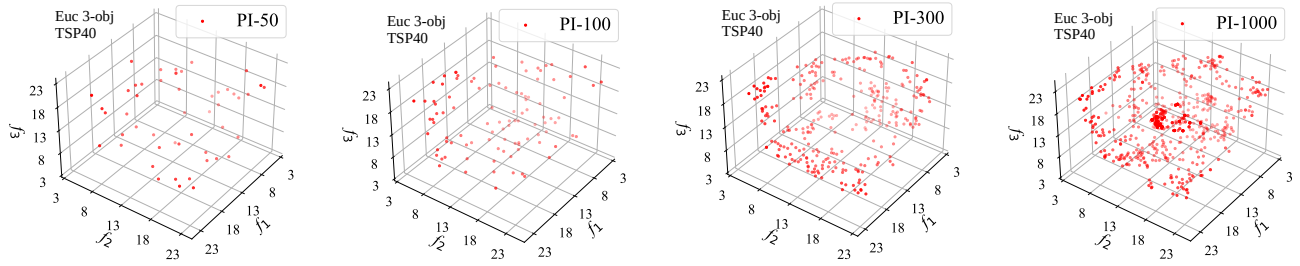
(a) Non-dominated sets (b) Overall non-dominated set $\mathcal{P}^*$

Fig. 1. The non-dominated objective vectors obtained by all methods on a randomly generated Euclidean 3-objective TSP40 instance.

TABLE II
THE MEAN OF HV, $|\mathbb{NDS}|$, AND $C1_R$ VALUES ACHIEVED BY FOUR LEARNING-BASED ALGORITHMS ON 200 RANDOMLY GENERATED MOCVRP INSTANCES. THE BEST RESULTS ARE MARKED IN BOLD.

Methods	MOCVRP20			MOCVRP50			MOCVRP100		
	HV	$\|\mathbb{NDS}\|$	$C1_R$	HV	$\|\mathbb{NDS}\|$	$C1_R$	HV	$\|\mathbb{NDS}\|$	$C1_R$
DRL-MOA	0.2534	4.76	5.49%	0.1079	4.42	6.29%	0.0917	4.73	6.13%
MODRL/D-AM	0.8390	10.65	21.67%	0.9353	11.08	30.25%	**0.9043**	8.73	**41.57%**
PI(random)	0.9120	**12.59**	**44.00%**	0.7881	8.86	21.92%	0.6204	10.68	23.17%
PI(rollout)	**0.9217**	11.92	28.84%	**0.9412**	**11.78**	**41.54%**	0.8180	**12.71**	29.13%

III. MORE DISCUSSION

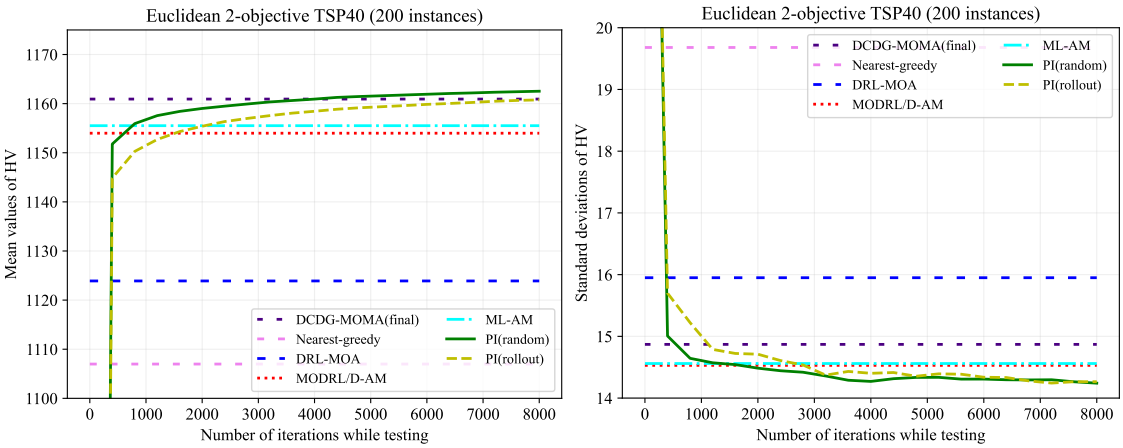


(a) $p_{size} = 50$ (b) $p_{size} = 100$ (c) $p_{size} = 300$ (d) $p_{size} = 1000$

Fig. 2. The non-dominated objective vectors obtained by PI(random) on a randomly generated Euclidean 3-objective TSP40 instances with different p_{size} values.

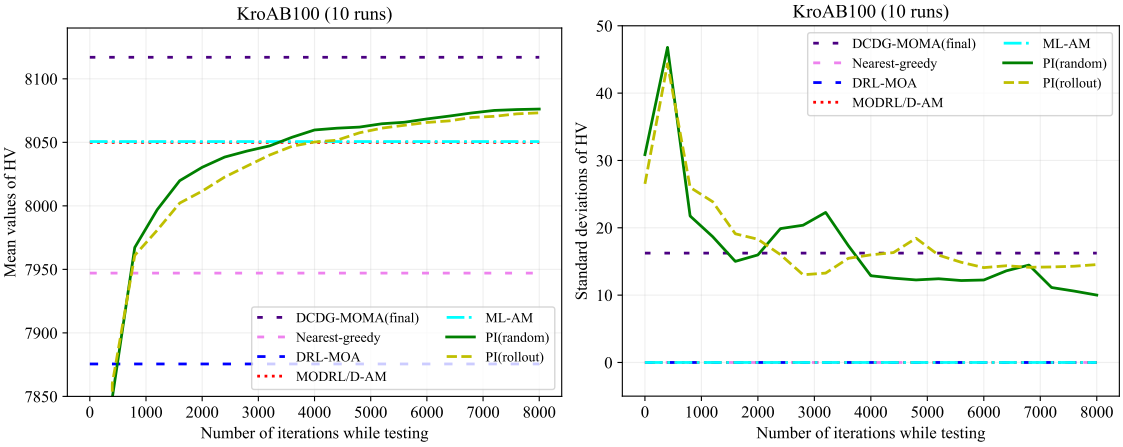
TABLE III
THE HV VALUE AND IMPROVEMENT RATE ACHIEVED BY THE TWO PI VARIANTS WITH DIFFERENT INITIAL SOLUTION POPULATIONS ON KROAB100, KROAB200, AND 200 EUCLIDEAN 3-OBJECTIVE TSP40 INSTANCES.

Methods	KroAB100		KroAB200		Euc 3-obj TSP40	
	HV	*Improvement rate*	HV	*Improvement rate*	HV	*Improvement rate*
Random initial population	2580.60	-	9552.38	-	10905.08	-
Random initial population + PI(random)	8076.39	212.97%	33891.19	254.79%	36596.78	235.59%
Random initial population + PI(rollout)	8073.23	212.84%	34048.60	256.44%	36758.31	237.08%
Nearest-greedy	7947.06	-	33813.18	-	33631.00	-
Initial population generated by Nearest-greedy + PI(random)	8117.45	2.14%	34073.41	0.77%	37133.39	10.41%
Initial population generated by Nearest-greedy + PI(rollout)	8123.91	2.23%	34316.23	1.49%	37290.34	10.88%
DRL-MOA	7875.53	-	33513.38	-	32860.33	-
Initial population generated by DRL-MOA + PI(random)	8118.73	3.09%	34071.47	1.67%	37085.29	12.86%
Initial population generated by DRL-MOA + PI(rollout)	8103.10	2.89%	34142.53	1.88%	37103.50	12.91%
MODRL/D-AM	8049.93	-	33972.43	-	36068.93	-
Initial population generated by MODRL/D-AM + PI(random)	8121.83	0.89%	34090.46	0.35%	37307.82	3.43%
Initial population generated by MODRL/D-AM + PI(rollout)	8115.32	0.81%	34142.12	0.50%	37309.21	3.44%



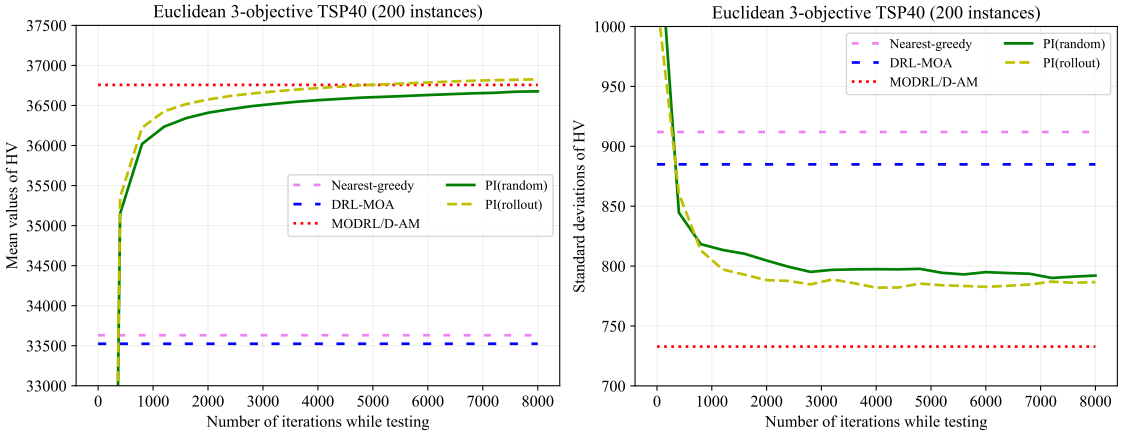
(a) Mean HV, Euc 2-objective TSP40

(b) Standard deviations of HV, Euc 2-objective TSP40



(c) Mean HV, KroAB100

(d) Standard deviations of HV, KroAB100



(e) Mean HV, Euc 3-objective TSP40

(f) Standard deviations of HV, Euc 3-objective TSP40

Fig. 3. The HV convergence curve along the number of iterations on the Euclidean 2-objective TSP40 dataset, KroAB100 instances, and the Euclidean 3-objective TSP40 dataset.

REFERENCES

[1] K. Li, T. Zhang, and R. Wang, "Deep reinforcement learning for multiobjective optimization," *IEEE Transactions on Cybernetics*, 2020.

[2] H. Wu, J. Wang, and Z. Zhang, "MODRL/D-AM: Multiobjective deep reinforcement learning algorithm using decomposition and attention model for multiobjective optimization," in *International Symposium on Intelligence Computation and Applications*. Springer, 2019, pp. 575–589.

[3] Z. Zhang, Z. Wu, H. Zhang, and J. Wang, "Meta-learning-based deep reinforcement learning for multiobjective optimization problems," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[4] Y. Shao, J. C.-W. Lin, G. Srivastava, D. Guo, H. Zhang, H. Yi, and A. Jolfaei, "Multi-objective neural evolutionary algorithm for combinatorial optimization problems," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

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

[6] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

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

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[9] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!" in *International Conference on Learning Representations*, 2018.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. pmlr, 2015, pp. 448–456.

[12] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *International Conference on Machine Learning*, 2010.

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

[14] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," *Advances in Neural information processing systems*, vol. 28, 2015.

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