Learning Encodings for Constructive Neural Combinatorial Optimization Needs to Regret

Rui Sun[1*], **Zhi Zheng**[1*], **Zhenkun Wang**[2†]

[1]Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China
[2]School of System Design and Intelligent Manufacturing and Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China
{sunr2020, zhengz2020}@mail.sustech.edu.cn, wangzhenkun90@gmail.com

Abstract

Deep-reinforcement-learning (DRL) based neural combinatorial optimization (NCO) methods have demonstrated efficiency without relying on the guidance of optimal solutions. As the most mainstream among them, the learning constructive heuristic (LCH) achieves high-quality solutions through a rapid autoregressive solution construction process. However, these LCH-based methods are deficient in convergency, and there is still a performance gap compared to the optimal. Intuitively, learning to regret some steps in the solution construction process is helpful to the training efficiency and network representations. This article proposes a novel regret-based mechanism for an advanced solution construction process. Our method can be applied as a plug-in to any existing LCH-based DRL-NCO method. Experimental results demonstrate the capability of our work to enhance the performance of various NCO models. Results also show that the proposed LCH-Regret outperforms the previous modification methods on several typical combinatorial optimization problems. The code and Supplementary File are available at https://github.com/SunnyR7/LCH-Regret.

Introduction

Many real-world applications involve combinatorial optimization problems (COPs) (Korte et al. 2011) like vehicle routing problems (Toth and Vigo 2002, 2014) and job scheduling problems (Taillard 1993). The optimization of combinatorial optimization problems is difficult due to its NP-hardness (Ausiello et al. 2012; Papadimitriou and Steiglitz 1998). In practice, it is crucial to obtain a near-optimal solution with less time and space consumption.

Neural combinatorial optimization (NCO) has recently demonstrated its superiority in dealing with various COPs (Kool, van Hoof, and Welling 2019; Drakulic et al. 2023; Luo et al. 2023; Huang et al. 2023; Li, Zhang, and Wang 2021; Zhang et al. 2023). Especially, deep reinforcement learning (DRL) based methods demonstrate excellent performance without utilizing the optimal solution as ground-truth labels for model training (Bello et al. 2017; Chen and Tian 2019; Wu et al. 2022; Ma et al. 2021; Wang et al. 2023). According to the differences in the generation process of solutions, DRL-NCO methods can be broadly classified into two categories: learning improvement heuristics (LIH) (Wu et al. 2022; Zheng et al. 2023) and learning constructive heuristics (LCH) (Kwon et al. 2020; Bengio, Lodi, and Prouvost 2021). Previous studies indicate that LCH methods develop faster inference speed and demonstrate advantages in generalization ability (Ma et al. 2021; Liu et al. 2023). The current network architecture of LCH-based DRL-NCO methods, such as Attention Model (AM) (Kool, van Hoof, and Welling 2019), Pointerformer (Jin et al. 2023), and POMO (Kwon et al. 2020), adopts a lightweight decoder and a heavy encoder with large numbers of parameters. Moreover, LCH methods always generate the solution sequentially in an autoregressive manner which selects a node at each step, and the node encodings obtained from the encoder are repeatedly involved in decoder calculations. Therefore, in the training process, it is significant to converge encoders faster and attain better encodings. However, the greedy decoding policy of the LCH-based DRL-NCO method is more likely to lead to a local optimum, making it difficult to achieve high-quality encodings (Kim, Park, and Park 2022; Yuan et al. 2023).

Recently, some methods have been proposed to modify the LCH-based DRL-NCO method. Some of them focus on designing a better decoder (Xin et al. 2021; Grinsztajn et al. 2023) while the others add auxiliary losses for some implicit regulations of the encodings (Kim, Park, and Park 2022; Yuan et al. 2023). These methods achieve a clear improvement in performance and the speed of convergence. Due to the significant difference in the parameter size between the encoder and the decoder, we can mainly attribute the improvement in performance to the optimization of the encodings. These modifications, however, remain the autoregressive solution construction process unchanged. It calls for a more effective solution construction process for better convergence.

In this work, we propose a regret-enabled LCH method named LCH-Regret to obtain better encodings. It contains a novel regret mechanism that allows rolling back to the previous node while constructing solutions step by step. In practice, we introduce a learnable vector termed Regret encoding to represent the features of the regret mechanism. Moreover, By learning a regret operation, LCH-Regret is the first method to modify the solution construction process. We demonstrate that this novel solution generation process contributes a lot to

[*]These authors contributed equally.
[†]Corresponding author.

faster convergence and better encodings. Without modifying the original model, LCH-Regret is model-agnostic, enabling it to augment the performance of any established, powerful LCH-based NCO model. In the experiment, we demonstrate that LCH-Regret is model-agnostic. Both AM+LCH-Regret (named AM-Regret), POMO+LCH-Regret (POMO-Regret), and MatNet+LCH-Regret (MatNet-Regret) narrow the optimal gap on various COPs, including the traveling salesman problem (TSP), capacitated vehicle routing problem (CVRP), and flexible flow shop problem (FFSP).

Our contributions can be summarized as follows:

- We propose a simple yet powerful LCH-Regret method which is the first to modify the autoregressive solution construction process.
- LCH-Regret adds a learnable vector to represent the regret embedding, and LCH-Regret is model-agnostic.
- POMO-Regret, AM-Regret, and MatNet-Regret demonstrate significantly better performance and generalization ability on several COPs.

Related Work

Deep Reinforcement Learning for Constructive Heuristics. Unlike the learning improvement heuristic that learns to execute operators to modify solutions iteratively, the learning constructive heuristic directly converts the input node coordinates into near-optimal solutions. Typically, the solutions are obtained with an autoregressive process, in which nodes are selected sequentially based on the current partial solutions (Vinyals, Fortunato, and Jaitly 2015). In Bello et al. (2017) and Nazari et al. (2018), the DRL technique (Williams 1992) is employed to train a constructive policy network for solving TSP and CVRP, respectively. Thereafter, by using the Transformer network (Vaswani et al. 2017), AM (Kool, van Hoof, and Welling 2019) and its improved version POMO (Kwon et al. 2020) achieve outstanding performances on various COPs. They employ a similar network structure that includes multiple multi-head attention encoder layers and a single-layer decoder. Compared to AM, POMO utilizes the solution symmetricity and proposes a multiple optima mechanism for a better reinforcement learning baseline. As a result, POMO achieves better node encodings and significantly outperforms AM.

Current Methods for Better Encodings. To narrow the optimal gap of LCH methods, several modification methods have already been proposed for LCH models. The multi-decoder AM (MDAM) (Xin et al. 2021) employs an ensemble of decoders and adds a Kullback-Leibler loss for the diversity of decoders. RL-CSL (Yuan et al. 2023) introduces an additional contrastive loss (Jaiswal et al. 2020), directly improving encodings' representation ability. It can achieve a better convergence of the AM. Sym-NCO (Kim, Park, and Park 2022) pays attention to the symmetric feature and leverages the symmetricity in vehicle routing problems. In detail, an auxiliary cosine similarity loss is added to obtain symmetric encodings. At the same time, they utilize the problem symmetricity to acquire a better baseline for the DRL training method, i.e., REINFORCE (Williams 1992).

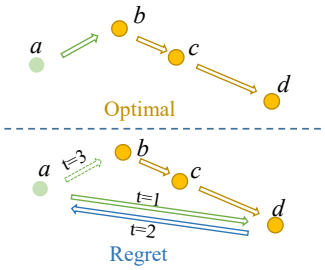


Figure 1: Regret mechanism allows policy network choosing to roll back to the previous step.

However, it should be noted that both the MDAM and RL-CSL necessitate additional architectural and hyperparameter design. Meanwhile, the symmetricity exploited by Sym-NCO can not apply to general COPs other than routing problems, such as the flexible flow shop problem (FFSP). Our LCH-Regret, however, is a model-agnostic approach that can adapt well to existing powerful LCH-based DRL-NCO models without redesigning the specific model structures. It can also be effectively applied to a broader range of COPs. Additionally, as demonstrated in experiments, LCH-Regret presents better performances than the aforementioned modification methods.

LCH-Regret

Learning Constructive Heuristics

Generally, the optimization object of COPs is a feasible solution $\boldsymbol{x} = (x_1, x_2, x_3, ...)$ where x_i represents the information of the i-th selection (e.g., a node index for vehicle routing problems). When constructing the feasible solution $\boldsymbol{x}$ step by step, LCH models are all based on the same Markov Decision Process (MDP) $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \boldsymbol{r}, \mathcal{P}\}$ (Kwon et al. 2020; Kool, van Hoof, and Welling 2019; Bello et al. 2017), which can be represented as follows:

State. The state represents the current partial solution. The state in the t-th time step is the current partial solution with t nodes $s_t = \boldsymbol{x}_t = (x_1, x_2, ..., x_t)$. s_0 is empty and $s_T = \boldsymbol{x}$.

Action. The action is to select a node at time step t, i.e., $a_t = x_{t+1}$. The chosen node needs to ensure that the partial solution $s_{t+1} = (x_1, x_2, ..., x_t, x_{t+1})$ is valid.

Reward. Every single time step has the same reward $r_t = -f(\boldsymbol{x})$ which is only calculated after a feasible solution $\boldsymbol{x}$ is constructed. The $f(\boldsymbol{x})$ is the objective function of the COP.

Policy. The policy network π_θ chooses actions in different time step t according to s_t. For a given instance c, the probability of a solution can be calculated as follows:

$$p_\theta(\boldsymbol{x}|c) = \prod_{t=0}^{T-1} \pi_\theta(a_t|s_t, c) = \prod_{t=1}^{T} \pi_\theta(x_t|\boldsymbol{x}_{1:t-1}, c), \quad (1)$$

where $\boldsymbol{x}_{1:t-1}$ is the partial solution from the first node to the $t-1$-th node in $\boldsymbol{x}$.

Regret Mechanism

Figure 1 gives an example of a regret operation used in the solution construction process. The regret operation at $t = 2$

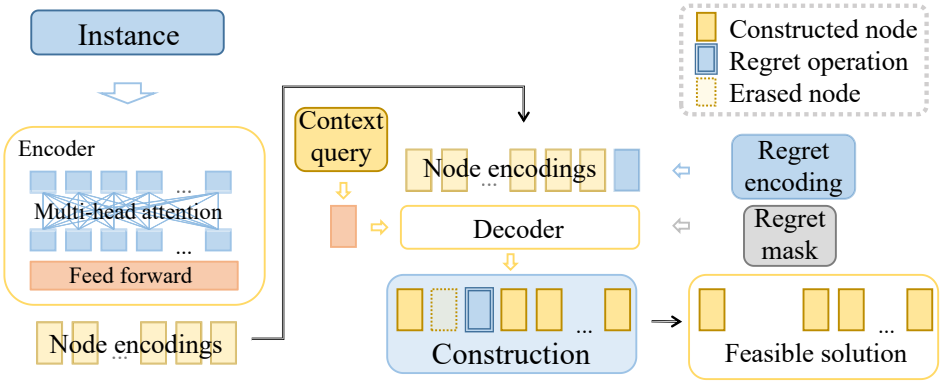


Figure 2: The modified network structure of the proposed LCH-Regret. LCH-Regret introduces an additional Regret encoding and Regret mask in a neural network for the regret mechanism. The Regret encoding serves as a token that lists parallel to node encodings, and the Regret mask is distinct for different situations and problems.

rolls back the selection of the previously selected node d, and the partial solution is subsequently erased from (a, d) to (a). Generally, the regret operation is defined as erasing the last node in a partial solution. LCH models will then use the information of the shortened partial solution for follow-up decodings.

Regret can also be modeled as a hierarchical control mechanism for partial solutions. But in this work, the regret operation serves as a virtual node parallel to candidate nodes. As shown in Figure 2, LCH-Regret introduces a learnable 128-dimensional Regret encoding and an additional Regret mask to the original LCH models. Due to the backpropagation mechanism, although the Regret encoding is structurally unaware of node encodings, it can still affect them in training. The Regret mask is maintained to avoid meaningless or illegal regret operations, and the detailed mask rules utilized in experiments are illustrated in Section Experiment Settings.

$\mathcal{L}_R$ for LCH-Regret Mechanism

To train policy networks for the regret mechanism, we introduce a modified MDP. The new MDP $\mathcal{M}' = \{\mathcal{S}', \mathcal{A}', \boldsymbol{r}, \mathcal{P}'\}$ of LCH-Regret is as follows:

New State. The new state $s'_t = \{\boldsymbol{x}_t, \mathbb{R}_t\} \in \mathcal{S}'$ now has two contents, the partial solution $\boldsymbol{x}_t = (x_1, ..., x_{|\boldsymbol{x}_t|})$ and regret records $\mathbb{R}_t$, where $|\boldsymbol{x}_t|$ is the length of the current solution. Due to regret operations, $|\boldsymbol{x}_t| \leq t$. $\mathbb{R}_t$ is a set of partial solutions and each element $\boldsymbol{r} \in \mathbb{R}_t$ represents a feasible partial solution where the policy network regrets the selection of the last node of this partial solution $\boldsymbol{r}$. s'_0 is empty and the final state $s'_T = \{\boldsymbol{x}_T, \mathbb{R}_T\}$ contains the complete solution $\boldsymbol{x}_T = (x_1, ..., x_{|\boldsymbol{x}_T|})$ and the set of all the regret records $\mathbb{R}_T$.

New Action. If the network chooses to construct a node a'_t, $s'_{t+1} = \{(\boldsymbol{x}_t, a'_t), \mathbb{R}_t\}$. But if the new action is the regret operation, a'_t = Regret, the new state will be $s'_{t+1} = \{\boldsymbol{x}_{t,1:|\boldsymbol{x}_t|-1}, \{\mathbb{R}_t, \boldsymbol{x}_t\}\}$. It means appending the partial solution to the set $\mathbb{R}_t$ and rolling back the last selection. $\boldsymbol{x}_{t,1:|\boldsymbol{x}_t|-1}$ represents the partial solution $\boldsymbol{x}_t$ without the last node. Additionally, Appendix B of the Supplementary File provides an example of the new MDP.

New Policy. Regret operations erase some selections in the construction process of solutions, so these nodes should not be involved in the policy calculation. For an instance c, the probability of a complete solution $\boldsymbol{x}$ is now as follows:

$$\begin{aligned} p'_\theta(\boldsymbol{x}, \mathbb{R}_T|c) &= \prod_{t=0}^{T-1} \pi_\theta(a'_t|\boldsymbol{x}_t, c) / \prod_{\boldsymbol{r}\in\mathbb{R}_T} \pi_\theta(r_{|\boldsymbol{r}|}|\boldsymbol{r}_{1:|\boldsymbol{r}|-1}, c) \\ &= \prod_{t=1}^{|\boldsymbol{x}|} \pi_\theta(x_t|\boldsymbol{x}_{1:t-1}, c) \prod_{\boldsymbol{r}\in\mathbb{R}_T} \pi_\theta(\text{Regret}|\boldsymbol{r}, c), \end{aligned} \tag{2}$$

where $\boldsymbol{r}_{1:|\boldsymbol{r}|-1}$ represents the partial solution $\boldsymbol{r}$ without the last node $r_{|\boldsymbol{r}|}$ and $\boldsymbol{x}_{1:t-1}$ is the partial solution from the first node to the $t-1$-th node in $\boldsymbol{x}$.

LCH-Regret also employs the REINFORCE gradient estimator (Williams 1992) to train the network. The gradient of the total training loss $\mathcal{L}$ is as follow:

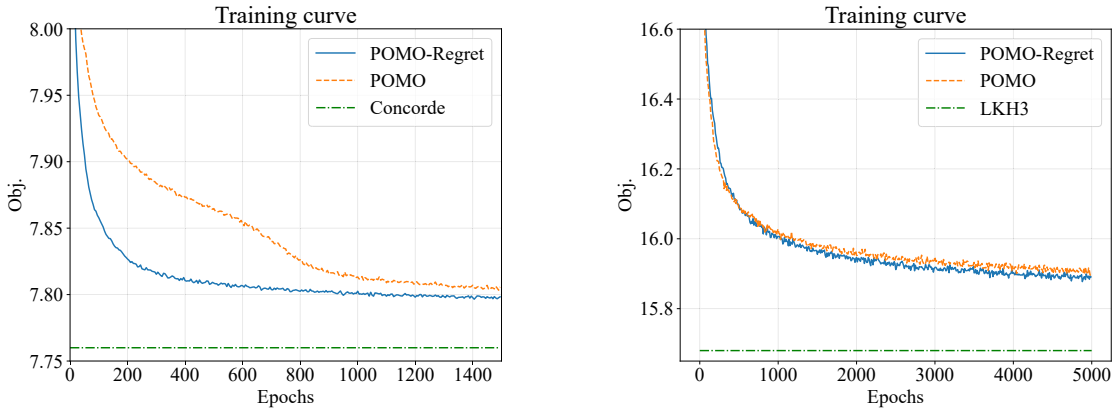
$$\begin{aligned} \nabla_\theta \mathcal{L}(c) = -\, &\mathbb{E}_{\boldsymbol{x},\mathbb{R}_T\sim p'_\theta(\cdot|c)} \\ &\Big[(f(\boldsymbol{x}) - b(c)) \nabla_\theta \log p'_\theta(\boldsymbol{x}, \mathbb{R}_T|c)\Big], \end{aligned} \tag{3}$$

where $b(c)$ is the baseline function. LCH-Regret being a model-agnostic method, the $\mathcal{L}$ can be easily obtained by introducing a novel loss $\mathcal{L}_R$ to the original training loss of LCH models $\mathcal{L}_{LCH}$:

$$\begin{aligned} \nabla_\theta \mathcal{L}_R(c) = -\, &\mathbb{E}_{\boldsymbol{x},\mathbb{R}_T\sim p'_\theta(\cdot|c)} \\ &\Big[(f(\boldsymbol{x}) - b(c)) \sum_{\boldsymbol{r}\in\mathbb{R}_T} \nabla_\theta \log \pi_\theta(\text{Regret}|\boldsymbol{r}, c)\Big], \end{aligned} \tag{4}$$

$$\begin{aligned} \nabla_\theta \mathcal{L}(c) &= \nabla_\theta \big[\mathcal{L}_{LCH}(c) + \mathcal{L}_R(c)\big] \\ &= -\, \mathbb{E}_{\boldsymbol{x},\mathbb{R}_T\sim p'_\theta(\cdot|c)} \Big[(f(\boldsymbol{x}) - b(c)) \nabla_\theta [\log p_\theta(\boldsymbol{x}|c) \\ &\quad + \sum_{\boldsymbol{r}\in\mathbb{R}_T} \log \pi_\theta(\text{Regret}|\boldsymbol{r}, c)]\Big], \end{aligned} \tag{5}$$

where $p_\theta(\boldsymbol{x})$ and $p'_\theta(\boldsymbol{x})$ are given in Eq. 1 and Eq. 2, respectively. By adding $\mathcal{L}_R$ to various $\mathcal{L}_{LCH}$, LCH-Regret can be applied to different LCH methods as a plug-in.



(a) POMO-Regret TSP100 (b) POMO-Regret CVRP100

Figure 3: Training curve of POMO-Regret on TSP and CVRP. The proposed LCH-Regret improves the convergence speed and final performance.

Experiment

In this section, we present the experimental results of LCH-Regret on a range of COPs, including TSP, CVRP, and FFSP. To demonstrate LCH-Regret's adaptability across different encoder-decoder-based NCO methods, we integrate it into two widely-used LCH-based DRL-NCO methods, POMO (Kwon et al. 2020) and AM (Kool, van Hoof, and Welling 2019), both for TSP and CVRP. Due to the inability of these two methods for FFSP, we integrate the regret mechanism into another NCO method MatNet (Kwon et al. 2021), which is the SOTA method for FFSP.

Problem Definition

TSP. Traveling salesman problem (TSP) is one of the most representative COPs. Let $\{C = c_{j,k}, j = 1, \ldots, n, k = 1 \ldots, n\}$ be the $n \times n$ cost metrics of a n nodes TSP instance, where $c_{j,k}$ denotes the cost between nodes j and k. The goal is to minimize the following equation:

$$f(\boldsymbol{x}) = \sum_{t=1}^{n-1} c_{x_t, x_{t+1}} + c_{x_n, x_1}, \tag{6}$$

where the solution $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ represents a permutation of all nodes, i.e., x_t is unique for $t = 1, 2, \ldots, n$.

The TSP instances used for training are randomly sampled coordinates that follow a uniform distribution. We also use TSP instances sampled from different distributions in the testing process. These instances of TSP are with "mixed" distribution or "clustered" distribution, which are proposed and tested in Jiang et al. (2022); Bi et al. (2022). We employ these instances for testing to assess the generalization ability of models in solving TSP instances with different distributions.

CVRP. We also test models on the capacitated vehicle routing problem (CVRP), which incorporates the vehicle capacity as a restriction while considering the shortest distance at the same time. The CVRP can be formalized as follows:

$$\begin{aligned} \text{minimize} \quad & f(\boldsymbol{x}) = \sum_{j=1}^{q} C(\boldsymbol{\rho}^j), \\ & C(\boldsymbol{\rho}^j) = \sum_{t=0}^{|\boldsymbol{\rho}^j|-1} c_{x_t, x_{t+1}} + c_{x_{n_j}, x_0}, \\ \text{subject to} \quad & 0 \le \delta_i \le D, \quad i = 1, \ldots, n, \\ & \sum_{i \in \boldsymbol{\rho}^j} \delta_i \le D, \quad j = 1, \ldots, q, \end{aligned} \tag{7}$$

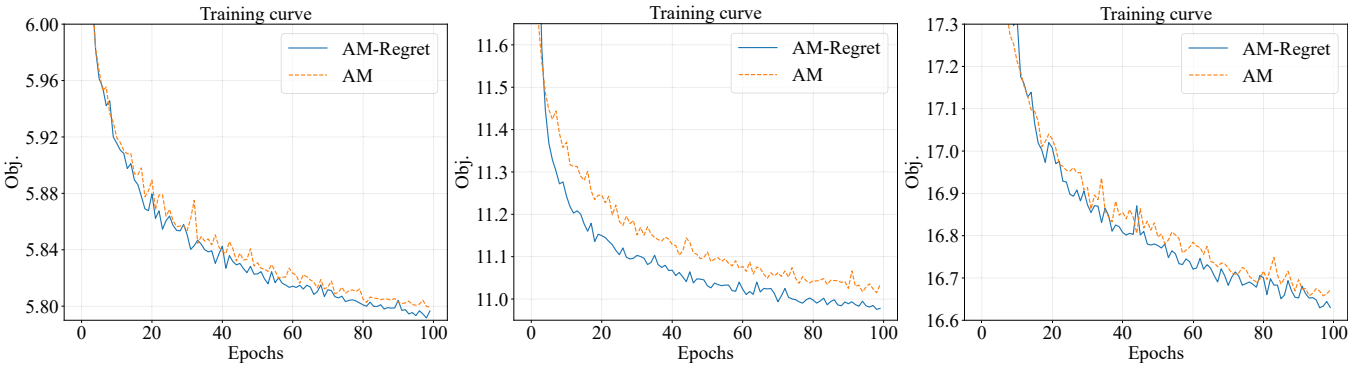
where $\boldsymbol{x}$ is the complete route of the vehicle which consists of q sub-routes $\{\boldsymbol{\rho}^1, \boldsymbol{\rho}^2, \ldots, \boldsymbol{\rho}^q\}$. Each sub-route starts from the depot x_0 and backs to x_0 and n_j represents the number of customers in it. $n = \sum_{j=1}^{q} n_j$ is the total number of customers. δ_i denotes the demand of node i and D denotes the capacity of the vehicle. Each customer is only allowed to visit once in the complete route $\boldsymbol{x}$. For CVRP, we solve the problem prescribed by Kool, van Hoof, and Welling (2019).

FFSP. The flexible flow shop problem (FFSP) is also introduced to demonstrate the ability of LCH-Regret to handle production scheduling problems. It is typically used to simulate production scheduling processes in real manufacturing applications. An FFSP instance considers the scheduling process of n jobs on several stages and the goal is to minimize the total time consumption. The settings of FFSP are consistent with the MatNet (Kwon et al. 2021).

Experiment Setting

Regret Mask Setups. LCH-Regret employs one-step regret, which is the most straightforward regret operation. To ensure the solutions remain feasible after regret operations, as shown in Figure 2, we introduce the Regret mask additionally. Considering all special situations, the basic Regret mask is designed generally based on the following three rules.

- Temporarily mask the node erased by the regret operation of step t at step t+1.



(a) AM-Regret TSP50 (b) AM-Regret CVRP50 (c) AM-Regret CVRP100

Figure 4: Training curve of AM-Regret on TSP and CVRP. The Training curve of AM-Regret for TSP100 is in Appendix D of the Supplementary File.

- Mask the regret operation in the first two steps of solution generation.
- Mask the regret operation in the two steps after executing each regret.

The first two criteria are designed to avoid unnecessary regret operations, while the last criterion ensures that the autoregressive decoding process does not get stuck in an infinite loop. The Regret mask is practically maintained through a finite state automaton described in Appendix B of the Supplementary File. Due to the different constraints of COPs, the Regret Mask may vary. Nevertheless, we believe the basic Regret mask we adopted can work for a broad range. In experiments, we utilize the same basic Regret mask for solving TSP and FFSP. For CVRP, additionally, particular rules are illustrated in Appendix B of the Supplementary File.

Training Setups. In our work, the proposed LCH-Regret is applied to the Attention Model (AM) (Kool, van Hoof, and Welling 2019), POMO (Kwon et al. 2020), and MatNet (Kwon et al. 2021), and the proposed models are named AM-Regret, POMO-Regret, and MatNet-Regret, respectively. Models are trained on a single Tesla-V100 GPU with the same hyperparameters (e.g., the training epoch and learning rate) as the original model. Figure 3 and 4 are the training curves of POMO-Regret and AM-Regret on various TSP and CVRP problems where the Obj. in plots represents the objective value during training. On all of these testing problems, LCH-Regret significantly improves the performance and in some of them, the training process is also sped up. Appendix D of the Supplementary File provides the training curves of other random seeds and the training time.

Testing Setups. For all the TSP, CVRP, and FFSP, the times the policy network operates to regret will gradually decrease after the network has sufficient confidence in its construction choice. Therefore, in these problems, we utilize the version that does not allow any regret operations in testing, which is considered the default due to a better generalization ability. Moreover, we also adopt efficient data-augmentation methods proposed in POMO (Kwon et al. 2020) ($\times$8 Augment) and MatNet (Kwon et al. 2021) ($\times$128 Augment) in the testing phase.

Results on the TSP

Algorithms are investigated on several randomly generated test sets of different scales and distributions and the TSPlib (Reinelt 1991). We compare the proposed AM-Regret and POMO-Regret with an exact solver Concorde (Applegate et al. 2006), heuristic algorithm LKH3 (Lin and Kernighan 1973), and OR Tools (Perron and Furnon 2023). LCH methods (AM (Kool, van Hoof, and Welling 2019), RL-CSL (Yuan et al. 2023), POMO (Kwon et al. 2020), Sym-NCO (Kim, Park, and Park 2022)) are also employed as baselines.

Following the settings of other LCH methods, we train two models separately for TSP instances with 50-node and 100-node (called TSP50/TSP100, respectively). Table 1 shows the in-domain performance of LCH-Regret, the proposed POMO-Regret with $\times$8 Augment (POMO-Regret-8 aug in Table) demonstrates the best result compared to all the learning-based baselines. Compared to AM and POMO, the proposed AM-Regret and POMO-Regret show significant improvements. Moreover, our LCH-Regret demonstrates better performances compared to previous modification methods like RL-CSL and Sym-NCO. It is worth noting that LCH-Regret models slightly increase the running time, which is mainly due to the consumption of maintaining the Regret mask.

Generalization Test. The generalization ability is also an important concern for the learning-based NCO solver (Joshi et al. 2022). Real-world instances may vary in size and distribution, and the solver has to generate good solutions for the instances unseen in training. This is essential to ensure the solver's practical applicability and usefulness in real-world scenarios. Besides variation in instance scales, we conduct zero-shot generalization tests on several distributions, including the basic uniform distribution, the "mixed" distribution, and the "clustered" distribution (Jiang et al. 2022).

Table 2 shows the optimal gap to the Concorde exact TSP solver (Applegate et al. 2006) on nine datasets of different scales and distributions. Results demonstrate that compared to POMO, and Sym-NCO, the proposed POMO-Regret significantly narrows the gaps in all nine datasets. Zhou et al. (2023) provides more datasets for generalization tests and in terms of the results in Appendix D of the Supplementary

Methods	TSP50			TSP100			CVRP50			CVRP100		
	Cost	Gap	Time	Cost	Gap	Time	Cost	Gap	Time	Cost	Gap	Time
Concorde(TSP)/HGS(CVRP)	5.6903	0.00%	13m	7.7609	0.00%	60m	10.3659	0.00%	2h	15.5757	0.00%	3.5h
LKH3	5.6903	0.00%	6m	7.7609	0.00%	25m	10.3670	0.10%	1h	15.6673	0.59%	2h
OR Tools	5.8447	2.71%	10m	8.0478	3.70%	25m	11.0743	6.83%	18m	17.2296	10.62%	69m
Attention Model(AM)	5.8003	1.93%	2s	8.1029	4.41%	5s	11.0330	6.44%	2s	16.7566	7.58%	5s
RL-CSL	5.7862	1.69%	2s	8.0994	4.36%	5s	10.9378	5.52%	2s	16.7411	7.50%	5s
AM-Regret	5.7946	1.83%	6s	8.0927	4.33%	13s	10.9785	5.91%	3s	16.6543	6.92%	6s
POMO	5.6972	0.12%	2s	7.7897	0.37%	9s	10.5524	1.80%	3s	15.9136	2.17%	12s
POMO-8 aug	5.6919	0.03%	14s	7.7711	0.13%	86s	10.4577	0.89%	18s	15.7863	1.35%	88s
Sym-NCO	-	-	-	7.7909	0.39%	9s	-	-	-	15.8853	1.99%	12s
Sym-NCO-8 aug	-	-	-	7.7724	0.15%	86s	-	-	-	15.7883	1.36%	88s
POMO-Regret	5.6970	0.12%	2s	7.7865	0.33%	10s	10.5450	1.73%	3s	15.8990	2.08%	13s
POMO-Regret-8 aug	**5.6916**	**0.02%**	15s	**7.7697**	**0.11%**	88s	**10.4526**	**0.84%**	19s	**15.7755**	**1.28%**	96s

Table 1: The average solution costs (Cost), average optimal gaps (Gap), and total inference times (Time) on 10,000 uniform TSP instances and 10,000 CVRP instances. The $\times 8$ Augment is abbreviated to 8 aug and the overall best results are in bold.

Methods	TSP100			TSP150			TSP200		
	Uniform	Cluster	Mixed	Uniform	Cluster	Mixed	Uniform	Cluster	Mixed
POMO	0.37%	3.33%	1.83%	0.89%	5.76%	3.89%	2.16%	9.83%	7.17%
Sym-NCO	0.39%	3.91%	1.46%	0.95%	5.61%	2.88%	2.25%	8.72%	5.12%
POMO-Regret	**0.33%**	**3.01%**	**1.41%**	**0.81%**	**4.91%**	**2.75%**	**1.93%**	**7.92%**	**4.71%**

Table 2: Generalization tests of models trained on uniform TSP100 instances. Each of the 9 datasets contains 10,000 randomly generated instances. The best result in each dataset is highlighted in bold.

File, POMO-Regret significantly outperforms as well. Overall, experimental results indicate that LCH-Regret not only benefits convergence but also enhances generalization ability. Moreover, the outstanding performance on zero-shot generalization tasks proves that the node encodings trained with the proposed LCH-Regret are more representative.

TSPlib Benchmark. We also test our POMO-Regret on the widely-used (Kim, Park, and Park 2022) TSPlib benchmark (Reinelt 1991) which contains real-world instances from dramatically different scales and distributions. As to Table 3, POMO-Regret has a better average optimal gap (named Gap in Table). The detailed results for each instance can be found in Appendix D of the Supplementary File.

Methods	Gap
POMO	5.03%
Sym-NCO	4.49%
POMO-Regret	**4.45%**

Table 3: Experimental results on TSPlib.

Results on CVRP

For CVRP, we compare all the available methods mentioned in TSP experiments besides introducing a strong Heuristic-based methods HGS (Vidal et al. 2012; Vidal 2022). Similar to TSP, we separately train two different models for CVRP instances with 50-customer and 100-customer. As shown in Table 1, POMO-Regret with $\times 8$ augment (POMO-Regret-8 aug in Table) attains the best results on both datasets. Meanwhile, similar to what is corroborated in TSP results, the AM-Regret and POMO-Regret demonstrate a significant improvement to the original AM and POMO, respectively. Results indicate that the LCH-Regret achieves better effects compared to other modifications of the corresponding LCH-based NCO methods on various problems.

Results on FFSP

Due to the inability of AM and POMO for FFSP, we integrate the regret mechanism into another LCH-based DRL-NCO method MatNet (Kwon et al. 2021), which is the SOTA method for FFSP. We conducted experiments on FFSP20, FFSP50, and FFSP100 with the same baseline algorithms and test sets provided in Kwon et al. (2021). Shortest Job First is a greedy heuristic. Both Genetic Algorithm (Kahraman et al. 2008) and Particle Swarm Opt. (Singh and Mahapatra 2012) are metaheuristics. Experimental results are shown in Table 4. As to the results, MarNet-Regret with $\times 128$ Augment stably outperforms, which means that the proposed LCH-Regret is still valid in FFSP. More details of baseline algorithms and training curves are provided in Appendix D of the Supplementary File.

In conclusion, the proposed LCH-Regret, especially the POMO-Regret and MatNet-Regret, demonstrates outstanding in-domain performance and generalization ability on three COPs. LCH-Regret reduces the optimal gap of the original LCH model on a total of 17 datasets and significantly defeats the current best modification method Sym-NCO. In addition, the successful applications of LCH-Regret to various models for solving TSP, CVRP, and FFSP corroborate that our LCH-Regret is a model-agnostic plug-in method to upgrade the performance of an arbitrary LCH-based DRL-NCO method.

Methods	FFSP20			FFSP50			FFSP100		
	Cost	Gap	Time	Cost	Gap	Time	Cost	Gap	Time
Shortest Job First	31.3	6.0	40s	57.0	7.6	1m	99.3	10.0	2m
Genetic Algorithm	30.6	5.3	7h	56.4	7.0	16h	98.7	9.4	29h
Particle Swarm Opt.	29.1	3.8	13h	55.1	5.7	26h	97.3	8.0	48h
MatNet	27.3	2.0	8s	51.5	2.1	14s	91.5	2.2	27s
MatNet-128 aug	25.4	0.1	3m	49.6	0.2	8m	89.7	0.4	23m
MatNet-Regret	27.2	1.9	8s	51.2	1.8	15s	91.1	1.8	31s
MatNet-Regret-128 aug	**25.3**	-	3m	**49.4**	-	8m	**89.3**	-	24m

Table 4: The average solution costs (Cost) and total inference times (Time) on 1,000 FFSP instances. The Gap in the Table measures the difference from the best result. The $\times 128$ Augment is abbreviated to 128 aug and the best results are in bold.

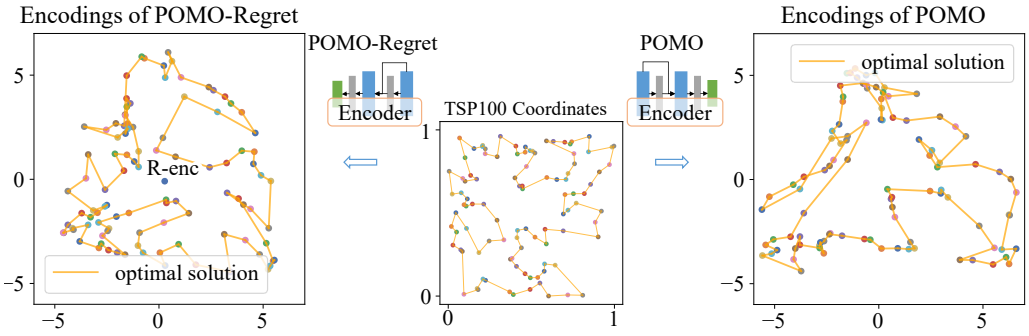


Figure 5: Principal component analysis (PCA) of node encodings encoded from the same TSP100 instance. R-enc represents the Regret encoding. Compared to POMO, after the same training steps, POMO-Regret achieves more uniform and sparse encodings.

Discussion

Better Node Encodings versus Better Decoding

The experimental results from our study show that the proposed LCH-Regret significantly enhances the original LCH models. Due to LCH models employing the structure with a heavy encoder and a light decoder (Kwon et al. 2020; Kool, van Hoof, and Welling 2019) and the significance of node encodings in the solution construction process, we can attribute this improvement to better node encodings achieved by our LCH-Regret compared to original LCH models.

To intuitively check if node encodings are refined, we use principal component analysis (PCA) (Maćkiewicz and Ratajczak 1993) to visualize the top 2 components of 128-dimensional node encodings of POMO-Regret. Besides, we compare the PCA results of POMO (Kwon et al. 2020) and the proposed POMO-Regret. Figure 5 shows the analysis result with a random TSP100 instance as input. The POMO and POMO-Regret models generating the encodings are both trained for 500 epochs. To better illustrate the difference between the encodings learned by the two different models, we use the optimal solution obtained by the Concorde algorithm (Applegate et al. 2006) to link each point in each subfigure (i.e., yellow lines). It demonstrates that, with the help of the regret mechanism, the node encodings of POMO-Regret are more uniform and sparse, which may make it easier to decode the optimal solution. The advantage of node encodings is not accidental and we provide more plots in Appendix D of the Supplementary File to further demonstrate the superiority.

Effectiveness of $\nabla_\theta \mathcal{L}_R$

The additional loss $\nabla_\theta \mathcal{L}_R$ can also efficiently lead to better node encodings and a more advanced training process with the help of the Regret encoding. In the training process of original LCH models, all nodes in a solution are given the same reward, which may lead to falling into local optimum (Sutton and Barto 2018). However, after the convergence of Regret encoding, with the guidance of $\nabla_\theta \mathcal{L}_R$, the Regret encoding can adaptively perceive whether a selected node is relatively acceptable, thus effectively optimizing the node encodings. This process assigns different gradients to different nodes in the same solution, which can also improve the convergence of LCH methods. We provide a theoretical explanation of this process for both TSP and CVRP in Appendix C of the Supplementary File.

Conclusion

In this paper, we propose a novel LCH-Regret method as a model-agnostic modification for solving different combinatorial optimization problems. It leverages a regret mechanism and is the first work to modify the solution construction process for better convergence and node encodings. Experiments on TSP, CVRP, and FFSP have demonstrated that our method is efficient for different COPs and significantly upgrades various LCH models including AM, POMO, and MatNet. This work provides valuable insights and inspires more prospects to explore the solution generation process of NCO models. In the future, we will try to explore better mechanisms for learning-based solvers of COPs.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No. 62106096), Characteristic Innovation Project of Colleges and Universities in Guangdong Province, China (Grant No. 2022KTSCX110), Shenzhen Technology Plan, China (Grant No. JCYJ20220530113013031), and Special Funds for the Cultivation of Guangdong College Students' Scientific and Technological Innovation, China ("Climbing Program" Special Funds)(Grant No. pdjh2024c21606).

References

Applegate, D.; Bixby, R.; Chvatal, V.; and Cook, W. 2006. Concorde TSP solver. https://www.math.uwaterloo.ca/tsp/concorde/index.html. Accessed: 2023-12-19.

Ausiello, G.; Crescenzi, P.; Gambosi, G.; Kann, V.; Marchetti-Spaccamela, A.; and Protasi, M. 2012. *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Springer Science & Business Media.

Bello, I.; Pham, H.; Le, Q. V.; Norouzi, M.; and Bengio, S. 2017. Neural Combinatorial Optimization with Reinforcement Learning. arXiv:1611.09940.

Bengio, Y.; Lodi, A.; and Prouvost, A. 2021. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2): 405–421.

Bi, J.; Ma, Y.; Wang, J.; Cao, Z.; Chen, J.; Sun, Y.; and Chee, Y. M. 2022. Learning Generalizable Models for Vehicle Routing Problems via Knowledge Distillation. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems*, volume 35, 31226–31238. Curran Associates, Inc.

Chen, X.; and Tian, Y. 2019. Learning to Perform Local Rewriting for Combinatorial Optimization. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Drakulic, D.; Michel, S.; Mai, F.; Sors, A.; and Andreoli, J.-M. 2023. BQ-NCO: Bisimulation Quotienting for Efficient Neural Combinatorial Optimization. arXiv:2301.03313.

Grinsztajn, N.; Furelos-Blanco, D.; Surana, S.; Bonnet, C.; and Barrett, T. D. 2023. Winner Takes It All: Training Performant RL Populations for Combinatorial Optimization. arXiv:2210.03475.

Huang, T.; Ferber, A. M.; Tian, Y.; Dilkina, B.; and Steiner, B. 2023. Searching Large Neighborhoods for Integer Linear Programs with Contrastive Learning. In Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S.; and Scarlett, J., eds., *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, 13869–13890. PMLR.

Jaiswal, A.; Babu, A. R.; Zadeh, M. Z.; Banerjee, D.; and Makedon, F. 2020. A survey on contrastive self-supervised learning. *Technologies*, 9(1): 2.

Jiang, Y.; Wu, Y.; Cao, Z.; and Zhang, J. 2022. Learning to solve routing problems via distributionally robust optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 9786–9794.

Jin, Y.; Ding, Y.; Pan, X.; He, K.; Zhao, L.; Qin, T.; Song, L.; and Bian, J. 2023. Pointerformer: Deep Reinforced Multi-Pointer Transformer for the Traveling Salesman Problem. arXiv:2304.09407.

Joshi, C. K.; Cappart, Q.; Rousseau, L.-M.; and Laurent, T. 2022. Learning the travelling salesperson problem requires rethinking generalization. *Constraints*, 27(1-2): 70–98.

Kahraman, C.; Engin, O.; Kaya, I.; and Kerim Yilmaz, M. 2008. An application of effective genetic algorithms for solving hybrid flow shop scheduling problems. *International Journal of Computational Intelligence Systems*, 1(2): 134–147.

Kim, M.; Park, J.; and Park, J. 2022. Sym-NCO: Leveraging Symmetricity for Neural Combinatorial Optimization. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems*, volume 35, 1936–1949. Curran Associates, Inc.

Kool, W.; van Hoof, H.; and Welling, M. 2019. Attention, Learn to Solve Routing Problems! arXiv:1803.08475.

Korte, B. H.; Vygen, J.; Korte, B.; and Vygen, J. 2011. *Combinatorial optimization*, volume 1. Springer.

Kwon, Y.-D.; Choo, J.; Kim, B.; Yoon, I.; Gwon, Y.; and Min, S. 2020. POMO: Policy Optimization with Multiple Optima for Reinforcement Learning. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 21188–21198. Curran Associates, Inc.

Kwon, Y.-D.; Choo, J.; Yoon, I.; Park, M.; Park, D.; and Gwon, Y. 2021. Matrix encoding networks for neural combinatorial optimization. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*, volume 34, 5138–5149. Curran Associates, Inc.

Li, K.; Zhang, T.; and Wang, R. 2021. Deep Reinforcement Learning for Multiobjective Optimization. *IEEE Transactions on Cybernetics*, 51(6): 3103–3114.

Lin, S.; and Kernighan, B. W. 1973. An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, 21(2): 498–516.

Liu, S.; Zhang, Y.; Tang, K.; and Yao, X. 2023. How Good Is Neural Combinatorial Optimization? A Systematic Evaluation on the Traveling Salesman Problem. arXiv:2209.10913.

Luo, F.; Lin, X.; Liu, F.; Zhang, Q.; and Wang, Z. 2023. Neural Combinatorial Optimization with Heavy Decoder: Toward Large Scale Generalization. arXiv:2310.07985.

Ma, Y.; Li, J.; Cao, Z.; Song, W.; Zhang, L.; Chen, Z.; and Tang, J. 2021. Learning to Iteratively Solve Routing Problems with Dual-Aspect Collaborative Transformer. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*, volume 34, 11096–11107. Curran Associates, Inc.

Maćkiewicz, A.; and Ratajczak, W. 1993. Principal components analysis (PCA). *Computers & Geosciences*, 19(3): 303–342.

Nazari, M.; Oroojlooy, A.; Snyder, L.; and Takac, M. 2018. Reinforcement Learning for Solving the Vehicle Routing Problem. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Papadimitriou, C. H.; and Steiglitz, K. 1998. *Combinatorial optimization: algorithms and complexity*. Courier Corporation.

Perron, L.; and Furnon, V. 2023. OR Tools. https://developers.google.com/optimization/. Accessed: 2023-12-19.

Reinelt, G. 1991. TSPLIB—A traveling salesman problem library. *ORSA journal on computing*, 3(4): 376–384.

Singh, M. R.; and Mahapatra, S. 2012. A swarm optimization approach for flexible flow shop scheduling with multiprocessor tasks. *The International Journal of Advanced Manufacturing Technology*, 62: 267–277.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Taillard, E. 1993. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2): 278–285.

Toth, P.; and Vigo, D. 2002. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics.

Toth, P.; and Vigo, D. 2014. *Vehicle routing: problems, methods, and applications*. Society for Industrial and Applied Mathematics.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is All you Need. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Vidal, T. 2022. Hybrid genetic search for the CVRP: Open-source implementation and SWAP* neighborhood. *Computers & Operations Research*, 140: 105643.

Vidal, T.; Crainic, T. G.; Gendreau, M.; Lahrichi, N.; and Rei, W. 2012. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3): 611–624.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer Networks. In Cortes, C.; Lawrence, N.; Lee, D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Wang, Z.; Yao, S.; Li, G.; and Zhang, Q. 2023. Multiobjective Combinatorial Optimization Using a Single Deep Reinforcement Learning Model. *IEEE Transactions on Cybernetics*.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3): 229–256.

Wu, Y.; Song, W.; Cao, Z.; Zhang, J.; and Lim, A. 2022. Learning Improvement Heuristics for Solving Routing Problems. *IEEE Transactions on Neural Networks and Learning Systems*, 33(9): 5057–5069.

Xin, L.; Song, W.; Cao, Z.; and Zhang, J. 2021. Multi-decoder attention model with embedding glimpse for solving vehicle routing problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 12042–12049.

Yuan, Z.; Li, G.; Wang, Z.; Sun, J.; and Cheng, R. 2023. RL-CSL: A Combinatorial Optimization Method Using Reinforcement Learning and Contrastive Self-Supervised Learning. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 7(4): 1010–1024.

Zhang, C.; Wu, Y.; Ma, Y.; Song, W.; Le, Z.; Cao, Z.; and Zhang, J. 2023. A review on learning to solve combinatorial optimisation problems in manufacturing. *IET Collaborative Intelligent Manufacturing*, 5(1): e12072.

Zheng, Z.; Yao, S.; Li, G.; Han, L.; and Wang, Z. 2023. Pareto Improver: Learning Improvement Heuristics for Multi-Objective Route Planning. *IEEE Transactions on Intelligent Transportation Systems*.

Zhou, J.; Wu, Y.; Song, W.; Cao, Z.; and Zhang, J. 2023. Towards Omni-generalizable Neural Methods for Vehicle Routing Problems. arXiv:2305.19587.