Sym-NCO: Leveraging Symmetricity for Neural Combinatorial Optimization

Minsu Kim **Junyoung Park** **Jinkyoo Park**
Korea Advanced Institute of Science and Technology (KAIST)
Dept. Industrial & Systems Engineering
`{min-su, Junyoungpark, jinkyoo.park}@kaist.ac.kr`

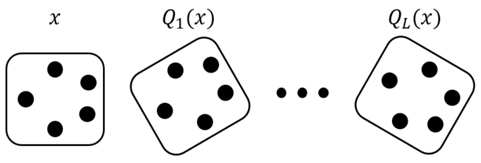
Abstract

Deep reinforcement learning (DRL)-based combinatorial optimization (CO) methods (i.e., DRL-NCO) have shown significant merit over the conventional CO solvers as DRL-NCO is capable of learning CO solvers less relying on problem-specific expert domain knowledge (heuristic method) and supervised labeled data (supervised learning method). This paper presents a novel training scheme, Sym-NCO, which is a regularizer-based training scheme that leverages universal symmetricities in various CO problems and solutions. Leveraging symmetricities such as rotational and reflectional invariance can greatly improve the generalization capability of DRL-NCO because it allows the learned solver to exploit the commonly shared symmetricities in the same CO problem class. Our experimental results verify that our Sym-NCO greatly improves the performance of DRL-NCO methods in four CO tasks, including the traveling salesman problem (TSP), capacitated vehicle routing problem (CVRP), prize collecting TSP (PCTSP), and orienteering problem (OP), without utilizing problem-specific expert domain knowledge. Remarkably, Sym-NCO outperformed not only the existing DRL-NCO methods but also a competitive conventional solver, the iterative local search (ILS), in PCTSP at 240$\times$ faster speed. Our source code is available at `https://github.com/alstn12088/Sym-NCO`.

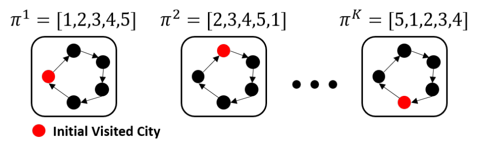
1 Introduction

Combinatorial optimization problems (COPs), mathematical optimization problems on discrete input space, have been used to solve numerous valuable applications, including vehicle routing problems (VRPs) [1, 2], drug discovery [3, 4], and semi-conductor design [5, 6, 7, 8, 9]. However, finding an optimal solution to COP is difficult due to its NP-hardness. Therefore, computing near-optimal solutions fast is essential from a practical point of view.

Conventionally, COPs are solved by integer program (IP) solvers or hand-crafted (meta) heuristics. Recent advances in computing infrastructures and deep learning have conceived the field of neural combinatorial optimization (NCO), a deep learning-based COP solving strategy. Depending on the training scheme, NCO methods are generally classified into supervised learning [10, 11, 12, 13, 14] and reinforcement learning (RL) [15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27]. Depending on the solution generation scheme, NCO methods are also classified into improvement [17, 16, 15, 28, 18, 19, 25] and constructive heuristics [20, 21, 22, 23, 24, 26, 27]. Among the NCO approaches, deep RL (DRL)-based constructive heuristics (i.e., DRL-NCO) are favored over conventional approaches for two major reasons. First, RL can be applied to train the NCO model in less explorered CO tasks because training RL does not require domain expert knowledge and supervised labels from a verified solver. Second, it is easy to produce qualified feasible solutions because the constructive process can easily avoid constraint-violated actions [21]. Despite the strength of DRL-NCO, there exists a performance gap between the state-of-the-art



(a) Problem symmetricity



(b) Solution symmetricity

Figure 2: Illustration of symmetricities in CO (exampled in TSP)

conventional heuristics and DRL-NCO. In an effort to close the gap, there have been attempts to employ problem-specific heuristics to existing DRL-NCO methods [23, 29]. However, devising a general training scheme to improve the performance of DRL-NCO still remains challenging. In this study, we propose the Symmetric Neural Combinatorial Optimization (Sym-NCO), a general training scheme applicable to universal CO problems. Sym-NCO is a regularization-based training scheme that leverages the symmetricities commonly found in COPs to increase the performance of existing DRL-NCO methods. Sym-NCO leverages two types of symmetricities innate in COP that are defined on the Euclidean graph. First, the problem symmetricity is derived from the rotational invariance of the solution; the rotated graph must exhibit the same optimal solution as the original graph as shown in Fig. 2a. Second, the solution symmetricity refers to the property that solutions have identical output values (See Fig. 2b).

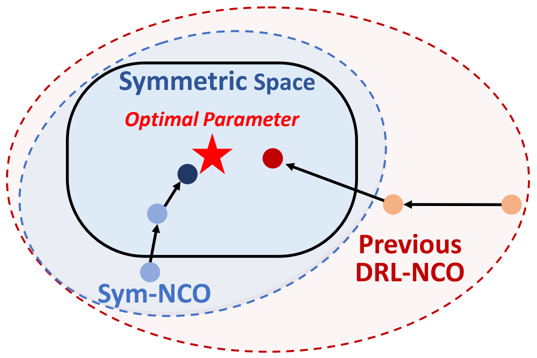


Figure 1: Illustration that describes an advantage of Sym-NCO. An optimal training parameter is in symmetric space. Sym-NCO makes a more compact training space compared with previous DRL-NCO and supports the NCO model efficiently converges in near-optimal parameters.

To train an effective NCO solver while leveraging the symmetricities, we employ REINFORCE algorithm with the baselines terms specially designed to impart solution and problem symmetricities. REINFORCE algorithm is used because the well-known effective NCO solvers are trained by REINFORCE; thus, we can improve such solvers by just modifying their baseline with our symmetricity-considered baseline terms. Specifically, we sample multiple solutions from the transformed problems and use the average return of them. Then, REINFORCE pushes each solution sampled by the solver to excel this baseline during training, thus improving the policy and making all the solutions the same, i.e., the problem and solution symmetricities are realized.

Motivation for learning symmetricity. Leveraging symmetricity is important to train CO models for two major reasons. Firstly, symmetricity is a strong inductive bias that can support the training process of DRL by making compact training space as shown in Fig. 1. Secondly, learning symmetricity is beneficial to increasing generalization capability for unseen CO problems because symmetricity induces the invariant representation that every COP contains.

Novelty. The major novelty of the proposed learning strategy is that it can easily improve existing powerful CO models; existing equivariant neural network schemes must be re-designed at the architecture level for adapting to CO.

2 Symmetricity in Combinatorial Optimization Markov Decision Process

This section presents several symmetric characteristics found in combinatorial optimization, which is formulated in the Markov decision process. The objective of NCO is to train the θ-parameterized solver F_θ by solving the following problem:

$$\theta^* = \arg\max_\theta \mathbb{E}_{\boldsymbol{P}\sim\rho}\left[\mathbb{E}_{\boldsymbol{\pi}\sim F_\theta(\boldsymbol{P})}\left[R(\boldsymbol{\pi};\boldsymbol{P})\right]\right] \tag{1}$$

where $\boldsymbol{P} = (\boldsymbol{x}, \boldsymbol{f})$ is a problem instance with the N node coordinates $\boldsymbol{x} = \{x_i\}_{i=1}^N$ and corresponding N features $\boldsymbol{f} = \{f_i\}_{i=1}^N$. The ρ is a problem generating distribution. The $\boldsymbol{\pi} = \{\pi_i\}_{i=1}^N$ is a solution where each element is index value $\pi_i \in \{1, ..., N\}$ for coordinates $\boldsymbol{x} = \{x_i\}_{i=1}^N$. The $R(\boldsymbol{\pi}; \boldsymbol{P})$ is objective value for $\boldsymbol{\pi}$ on problem $\boldsymbol{P}$.

For example, assume that we solve TSP with five cities (i.e. $N = 5$). Then the problem instance $\boldsymbol{P}$ contains five city coordinates $\boldsymbol{x} = \{x_i\}_{i=1}^5$ where the salesman must visit. The solution $\boldsymbol{\pi}$ is a sequence of city indices; if $\boldsymbol{\pi} = \{1, 3, 2, 5, 4\}$, the salesman visits city coordinates as $x_1 \to x_3 \to x_2 \to x_5 \to x_4 \to x_1$ (the salesman must go back to first visited city to complete a tour). In TSP, each city contains the homogeneous features $\boldsymbol{f}$. Thus, objective R of the TSP is defined as the negative of tour length: $R(\boldsymbol{\pi}; \boldsymbol{P}) = -\big(\sum_{i=1}^4 ||x_{\pi_{i+1}} - x_{\pi_i}|| + ||x_{\pi_5} - x_{\pi_1}||\big)$. This combinatorial decision process can be expressed as Markov decision process, and the solver $F_\theta(\boldsymbol{\pi}|\boldsymbol{P})$ can be expressed as *instance conditioned policy*. To this end, we can utilize deep reinforcement learning for training solver $F_\theta(\boldsymbol{\pi}|\boldsymbol{P})$. We formally define the Markov decision process for CO in the below chapter.

2.1 Combinatorial optimization Markov decision process

We define the combinatorial optimization Markov decision process (CO-MDP) as the sequential construction of a solution of COP. For a given $\boldsymbol{P}$, the components of the corresponding CO-MDP are defined as follows:

- **State.** The state $\boldsymbol{s}_t = (\boldsymbol{a}_{1:t}, \boldsymbol{x}, \boldsymbol{f})$ is the t-th (partially complete) solution, where $\boldsymbol{a}_{1:t}$ represents the previously selected nodes. The initial and terminal states $\boldsymbol{s}_0$ and $\boldsymbol{s}_T$ are equivalent to the empty and completed solution, respectively. In this paper, we denote the solution $\boldsymbol{\pi}(\boldsymbol{P})$ as the completed solution.
- **Action.** The action a_t is the selection of a node from the un-visited nodes (i.e., $a_t \in \mathbb{A}_t = \{\{1, ..., N\} \setminus \{\boldsymbol{a}_{1:t-1}\}\}$).
- **Reward.** The reward function $R(\boldsymbol{\pi}; \boldsymbol{P})$ maps the objective value from given $\boldsymbol{\pi}$ of problem $\boldsymbol{P}$. We assume that the reward is a function of $\boldsymbol{a}_{1:T}$ (solution sequences), $||x_i - x_j||_{i,j \in \{1,...N\}}$ (relative distances) and $\boldsymbol{f}$ (nodes features). In TSP, capacitated VRP (CVRP), and prize collecting TSPs (PCTSP), the reward is the negative of the tour length. In orienteering problem (OP), the reward is the sum of the prizes.

Having defined CO-MDP, we define the solution mapping as follows: $\boldsymbol{\pi} \sim F_\theta(\cdot|P) = \prod_{t=1}^T p_\theta(a_t|\boldsymbol{s}_t)$ where $p_\theta(a_t|\boldsymbol{s}_t)$ is the policy that produces a_t at $\boldsymbol{s}_t$, and T is the maximum number of states in the solution construction process.

2.2 Symmetricities in CO-MDP

Symmetricities are found in various COPs. We conjecture that imposing those symmetricities on F_θ improves the generalization and sample efficiency of F_θ. We define the two identified symmetricities that are commonly found in various COPs:

Definition 2.1 (**Problem Symmetricity**). Problem $\boldsymbol{P}^i$ and $\boldsymbol{P}^j$ are problem symmetric ($\boldsymbol{P}^i \overset{\text{sym}}{\longleftrightarrow} \boldsymbol{P}^j$) if their optimal solution sets are identical.

Definition 2.2 (**Solution Symmetricity**). Two solutions $\boldsymbol{\pi}^i$ and $\boldsymbol{\pi}^j$ are solution symmetric ($\boldsymbol{\pi}^i \overset{\text{sym}}{\longleftrightarrow} \boldsymbol{\pi}^j$) on problem $\boldsymbol{P}$ if $R(\boldsymbol{\pi}^i; \boldsymbol{P}) = R(\boldsymbol{\pi}^j; \boldsymbol{P})$.

An exemplary problem symmetricity found in various COPs is the rotational symmetricity:

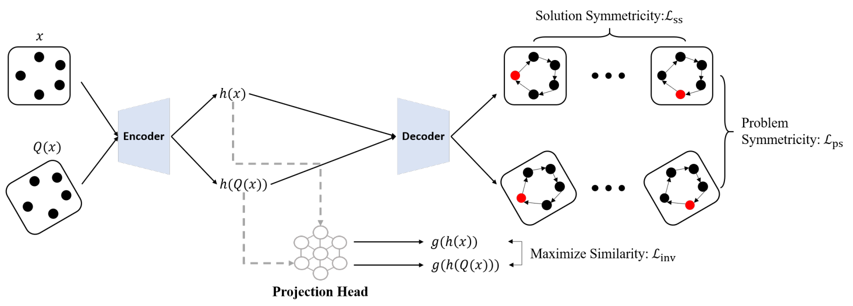
Theorem 2.1 (**Rotational symmetricity**). For any orthogoanl matrix Q, the problem $\boldsymbol{P}$ and $Q(\boldsymbol{P}) \triangleq \{\{Qx_i\}_{i=1}^N, \boldsymbol{f}\}$ are problem symmetric: i.e., $\boldsymbol{P} \overset{\text{sym}}{\longleftrightarrow} Q(\boldsymbol{P})$. See Appendix A for the proof.

Rotational problem symmetricity is identified in every Euclidean COPs. On the other hand, solution symmetricity cannot be identified easily as the properties of the solutions are distinct for every COP.

3 Symmetric Neural Combinatorial Optimization

This section presents Sym-NCO, an effective training scheme that leverages the symmetricities of COPs. Sym-NCO learns a solve F_θ by minimizing the total loss function:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{Sym-RL}} + \alpha\mathcal{L}_{\text{inv}} = \mathcal{L}_{\text{ps}} + \beta\mathcal{L}_{\text{ss}} + \alpha\mathcal{L}_{\text{inv}} \tag{2}$$



Solution Symmetricity: $\mathcal{L}_{ss}$
x
$Q(x)$
Encoder
$h(x)$
$h(Q(x))$
Decoder
Problem Symmetricity: $\mathcal{L}_{ps}$
Projection Head
$g(h(x))$
$g(h(Q(x)))$
Maximize Similarity: $\mathcal{L}_{inv}$

Figure 3: An overview of Sym-NCO

where $\mathcal{L}_{\text{Sym-RL}}$ is REINFORCE loss term supported by symmetricity and $\mathcal{L}_{\text{inv}}$ is the regularization loss to induce that invariant representation. The $\mathcal{L}_{\text{Sym-RL}}$ is composed with $\mathcal{L}_{\text{ss}}$, the REINFORCE loss term of Eq. (1) with solution symmetricity regularizing mechanism, and $\mathcal{L}_{\text{ps}}$, the REINFORCE loss term of Eq. (1) with both solution and problem symmetricity regularization. $\alpha, \beta \in [0, 1]$ are the weight coefficients. In the following subsections, we explain each loss term in detail.

3.1 Regularizing REINFORCE with problem and solution symmetricities via $\mathcal{L}_{\text{Sym-RL}}$

As discussed in Section 2.2, COPs have problem and solution symmetricities. We explain how to learn solver F_θ using REINFORCE with the specially designed baseline to approximately impose symmetricities. We provide the policy gradients to $\mathcal{L}_{\text{ss}}(\boldsymbol{\pi}(\boldsymbol{P}))$ and $\mathcal{L}_{\text{ps}}(\boldsymbol{\pi}(\boldsymbol{P}))$ in the context of the REINFORCE algorithm [30] with the proposed baseline schemes.

Leveraging solution symmetricity. As defined in Definition 2.2, the symmetric solutions must have the same objective values. We propose the REINFORCE loss $\mathcal{L}_{\text{ss}}$ with the baseline specially designed to exploit the solution symmetricity of CO as follows:

$$\mathcal{L}_{\text{ss}} = -\mathbb{E}_{\boldsymbol{\pi} \sim F_\theta(\cdot|\boldsymbol{P})}\left[R(\boldsymbol{\pi}; \boldsymbol{P})\right] \tag{3}$$

$$\nabla_\theta \mathcal{L}_{\text{ss}} = -\mathbb{E}_{\boldsymbol{\pi} \sim F_\theta(\cdot|\boldsymbol{P})}\Big[\big[R(\boldsymbol{\pi}; \boldsymbol{P}) - b(\boldsymbol{P})\big]\nabla_\theta \log F_\theta\Big] \tag{4}$$

$$\approx -\frac{1}{K}\sum_{k=1}^{K}\Big[\big[R(\boldsymbol{\pi}^k; \boldsymbol{P}) - \frac{1}{K}\sum_{k=1}^{K} R(\boldsymbol{\pi}^k; \boldsymbol{P})\big]\nabla_\theta \log F_\theta\Big] \tag{5}$$

where $\{\boldsymbol{\pi}^k\}_{k=1}^K$ are the solutions of P sampled from $F_\theta(\boldsymbol{\pi}|\boldsymbol{P})$, $\log F_\theta$ is the log-likelihood of F_θ, K is the number of sampled solutions, $b(\boldsymbol{P})$ is a shared baseline which is the average reward from K solutions for the identical problem $\boldsymbol{P}$.

The $\mathcal{L}_{\text{ss}}$ approximately imposes solution symmetricity using REINFORCE algorithm with a novel baseline $b(\boldsymbol{P})$. The sum of advantage in the solution group $\{\boldsymbol{\pi}^k\}_{k=1}^K$ is always zero:

$$\frac{1}{K}\sum_{k=1}^{K}\Big[\big[R(\boldsymbol{\pi}^k; \boldsymbol{P}) - \frac{1}{K}\sum_{k=1}^{K} R(\boldsymbol{\pi}^k; \boldsymbol{P})\big]\Big] = 0 \tag{6}$$

The $\mathcal{L}_{\text{ss}}$ induces competition among the rewards, $R(\boldsymbol{\pi}^1; \boldsymbol{P}), ..., R(\boldsymbol{\pi}^K; \boldsymbol{P})$ which can be seen as a zero-sum game. Therefore, $\mathcal{L}_{\text{ss}}$ improves the overall reward quality of the solution group using the proposed competitive REINFORCE scheme, making the solver generate high-rewarded solutions but a small reward deviation between solutions. The small reward-deviation indicates $\mathcal{L}_{\text{ss}}$ approximately imposes solution symmetricity to solver F_θ.

The POMO [23] employed a similar training technique with our $\mathcal{L}_{\text{ss}}$, that finds symmetric solutions by forcing F_θ to visit all possible initial cities when solving TSP and CVRP. However, the reward of COPs, including CVRP, PCTSP, and OP, is usually sensitive to first-city selection. Therefore, POMO can be an excellent standalone method for TSP but can be further improved using our $\mathcal{L}_{\text{ss}}$ loss term in other tasks including CVRP.

Leveraging problem symmetricity. As discussed in Section 2.2, the rotational problem symmetricity is common in various COPs. We propose the REINFORCE loss $\mathcal{L}_{\text{ps}}$ which is equipped with problem symmetricity:

$$\mathcal{L}_{\text{ps}} = -\mathbb{E}_{Q^l \sim \mathbf{Q}} \mathbb{E}_{\boldsymbol{\pi} \sim F_\theta(\cdot|Q^l(\boldsymbol{P}))} \left[R(\boldsymbol{\pi}; \boldsymbol{P}) \right] \tag{7}$$

$$\nabla_\theta \mathcal{L}_{\text{ps}} = -\mathbb{E}_{Q^l \sim \mathbf{Q}} \left[\mathbb{E}_{\boldsymbol{\pi} \sim F_\theta(\cdot|Q^l(\boldsymbol{P}))} \left[\left[R(\boldsymbol{\pi}; \boldsymbol{P}) - b(\boldsymbol{P}, \boldsymbol{Q}) \right] \nabla_\theta \log F_\theta \right] \right] \tag{8}$$

$$\approx \frac{1}{LK} \sum_{l=1}^{L} \sum_{k=1}^{K} \left[\left[R(\boldsymbol{\pi}^{l,k}; \boldsymbol{P}) - \frac{1}{LK} \sum_{l=1}^{L} \sum_{k=1}^{K} R(\boldsymbol{\pi}^{l,k}; \boldsymbol{P}) \right] \nabla_\theta \log F_\theta \right] \tag{9}$$

where $\mathbf{Q}$ is the distribution of random orthogonal matrices, Q^l is the l^{th} sampled rotation matrix, and $\boldsymbol{\pi}^{l,k}$ is the k^{th} sample solution of the l^{th} rotated problem. We construct L problem symmetric problems, $Q^1(\boldsymbol{P}), ..., Q^L(\boldsymbol{P})$, by using the sampled rotation matrices, and smaple K symmetric solutions from each of the L problems. Then, the shared baseline $b(\boldsymbol{P}, \boldsymbol{Q})$ is constructed by averaging $L \times K$ solutions.

Similar to the regularization scheme of $\mathcal{L}_{\text{ss}}$, the advantage term of $\mathcal{L}_{\text{ps}}$ also induces competition between solutions sampled from rotationally symmetric problems. Since the rotational symmetricity is defined such that x and $Q_l(x)$ have the same solution, the negative advantage value forces the solver to find a better solution. As mentioned in Section 2.2, problem symmetricity in COPs is usually pre-identified (i.e. there is provable guaranteed symmetricity such as rotational symmetricity Theorem 2.1); $\mathcal{L}_{\text{ps}}$ are applicable to general COPs. Moreover, multiple solutions are sampled for each symmetric problem so that $\mathcal{L}_{\text{ps}}$ can also identify and exploit the solution symmetricity with a similar approach taken for $\mathcal{L}_{\text{ss}}$. We provide detailed implementation and design guides regarding the integration strategy of $\mathcal{L}_{\text{ss}}$ and $\mathcal{L}_{\text{ps}}$ in Appendix C.2.

3.2 Learning invariant representation with Pre-identified Symmetricity: $\mathcal{L}_{\text{inv}}$.

By Theorem 2.1, the original problem x and its rotated problem $Q(x)$ have identical solutions. Therefore the encoder of F_θ can be enforced to have invariant representation by leveraging the pre-identified symmetricity: rotation symmetricity.

We denote $h(x)$ and $h\left(Q(x)\right)$ as the hidden representations of x and $Q(x)$, respectively. To impose the rotational invariant property on $h(x)$, we train solver F_θ with an additional regularization loss term $\mathcal{L}_{\text{inv}}$ defined as:

$$\mathcal{L}_{\text{inv}} = -S_{\cos}\left(g\Big(h(x)\Big), g\Big(h\big(Q(x)\big)\Big) \right) \tag{10}$$

where $S_{\cos}(a, b)$ is the cosine similarity between a and b. g is the MLP-parameterized projection head.

For learning the invariant representation on rotational symmetricity, we penalize the difference between the projected representation $g(h(x))$ and $g(h(Q(x)))$, instead of directly penalizing the difference between $h(x)$ and $h(Q(x)$. This penalizing scheme allows the use of an arbitrary encoder network architecture while maintaining the diversity of h [31]. We empirically verified that this approach attains stronger solvers as described in Section 6.1.

4 Related Works

Deep construction heuristics. Bello et al. [20] propose one of the earliest DRL-NCO methods, based on pointer network (PointerNet) [10], and trained it with an actor-critic method. Attention model (AM) [21] successfully extends [20] by swapping PointerNet with Transformer [32], and it is currently the *de-facto* standard method for NCO. Notably, AM verifies its problem agnosticism by solving several classical routing problems and their practical extensions [7, 19]. The multi-decoder AM (MDAM) [26] extends AM by employing an ensemble of decoders. However, such an extension is inapplicable for stochastic routing problems. The policy optimization for multiple optimal (POMO) [23] extends AM by exploiting the solution symmetricities in TSP and CVRP. Even though POMO shows significant improvements from AM, it relies on problem-specific solution symmetricities for TSP. Our method can be seen as a general-purpose symmetric learning scheme, extended from POMO, which can be applied in more general CO tasks.

Equivariant deep learning. In deep learning, symmetricities are often enforced by employing specific network architectures. Niu et al. [33] proposes a permutation equivariant graph neural network (GNN) that produces equivariant outputs to the input order permutations. The $SE(3)$-Transformer [34] restricts the Transformer so that it is equivariant to $SE(3)$ group input transformation. Similarly, equivariant GNN (EGNN) [35] proposes a GNN architecture that produces $O(n)$ group equivariant output. These network architectures can dramatically reduce the search space of the model parameters. Some research applies equivariant neural networks to RL tasks to improve sample efficiency [36]. Also, there are several works that exploited the symmetric nature of CO. Ouyang et al. [37] proposed equivariant encoding scheme by giving rule-based input transformation to input graph. Hudson et al. [38] suggested a line-graph embedding scheme, which is beneficial to process CO graphs with rotational equivariant. Our Sym-NCO is a regularization method that is capable of learning existing powerful CO models without giving rule-based hard constraints to the structure. We empirically study the benefit of Sym-NCO over other symmetricity-based approaches in Section 6.1 and Appendix D.5.

5 Experiments

This section provides the experimental results of Sym-NCO for TSP, CVRP, PCTSP, and OP. Focusing on the fact that Sym-NCO can be applied to any encoder-decoder-based NCO method, we implement Sym-NCO on top of POMO [23] to solve TSP and CVRP, and AM [21] to solve PCTSP and OP, respectively. We additionally validate the effectiveness of Sym-NCO on PointerNet [10] at TSP ($N = 100$).

5.1 Tasks and baseline selections

TSP aims to find the Hamiltonian cycle with a minimum tour length. We employ Concorde [39] and LKH-3 [40] as the non-learnable baselines, and PointerNet [10], the structured-to-vector deep-Q-network (S2V-DQN) [41], AM [21], POMO [23] and MDAM [26] as the neural constructive baselines.

CVRP is an extension of TSP that aims to find a set of tours with minimal total tour lengths while satisfying the capacity limits of the vehicles. We employ LKH-3 [40] as the non-learnable baselines, and Nazari et al. [22], AM [21], POMO [23] and MDAM [26] as the constructive neural baselines.

PCTSP is a variant of TSP that aims to find a tour with minimal tour length while satisfying the prize constraints. We employ the iterative local search (ILS) [21] as the non-learnable baseline, and AM [21] and MDAM [26] as the constructive neural baselines.

OP is a variant of TSP that aims to find the tour with maximal total prizes while satisfying the tour length constraint. We employ *compass* [42] as the non-learnable baseline, and AM [21] and MDAM [26] as the constructive neural baselines.

5.2 Experimental setting

Problem size. We provide the results of problems with $N = 100$ for the four problem classes, and real-world TSP problems with $50 < N < 250$ from TSPLIB.

Hyperparameters. We apply Sym-NCO to POMO, AM, and PointerNet. To make fair comparisons, we use the same network architectures and training-related hyperparameters from their original papers to train their Sym-NCO-augmented models. Please refer to Appendix Appendix C.1 for more details.

Dataset and Computing Resources. We use the benchmark dataset [21] to evaluate the performance of the solvers. To train the neural solvers, we use *Nvidia* A100 GPU. To evaluate the inference speed, we use an *Intel* Xeon E5-2630 CPU and *Nvidia* RTX2080Ti GPU to make fair comparisons with the existing methods as proposed in [26].

5.3 Performance metrics

This section provides detailed performance metrics:

Table 1: Performance evaluation results for TSP and CVRP. Bold represents the best performances in each task. '-' indicates that the solver does not support the problem. 's' indicates multi-start sampling, 'bs' indicates the beam search. '$\times 5$ for the MDAM indicates the 5 decoder ensemble.

Method		TSP ($N = 100$)			CVRP ($N = 100$)		
		Cost ↓	Gap	Time	Cost ↓	Gap	Time
Handcrafted Heuristic-based Classical Methods							
Concorde	Heuristic [39]	7.76	0.00%	3m		–	
LKH3	Heuristic [40]	7.76	0.00%	21m	15.65	0.00%	13h
RL-based Deep Constructive Heuristic methods with greedy rollout							
PointerNet $\{greedy.\}$	NIPS'15 [10, 20]	8.60	6.90 %	–		–	
S2V-DQN $\{greedy.\}$	NIPS'17 [41]	8.31	7.03 %	–		–	
RL $\{greedy.\}$	NeurIPS'18 [22]		–		17.23	10.12%	–
AM $\{greedy.\}$	ICLR'19 [21]	8.12	4.53%	2s	16.80	7.34%	3s
MDAM $\{greedy. \times 5\}$	AAAI'21 [25]	7.93	2.19%	36s	16.40	4.86%	45s
POMO $\{greedy.\}$	NeurIPS'20 [23]	7.85	1.04%	2s	16.26	3.93%	3s
Sym-NCO $\{greedy.\}$	*This work*	**7.84**	**0.94**%	2s	**16.10**	**2.88**%	3s
RL-based Deep Constructive Heuristic methods with multi-start rollout							
Nazari et al. $\{bs.10\}$	NeurIPS'18 [22]		–		16.96	8.39%	–
AM $\{s.1280\}$	ICLR'19 [21]	7.94	2.26%	41m	16.23	3.72%	54m
POMO {s. 100}	NeurIPS'20 [23]	7.80	0.44%	13s	15.90	1.67%	16s
MDAM {bs. 30 $\times$ 5}	AAAI'21 [25]	7.80	0.48%	20m	16.03	2.49%	1h
Sym-NCO $\{s.100\}$	*This work*	**7.79**	**0.39%**	13s	**15.87**	**1.46%**	16s

Table 2: Performance evaluation results for PCTSP and OP. Notations are the same with Table 1.

Method		PCTSP ($N = 100$)			OP ($N = 100$)		
		Cost ↓	Gap	Time	Obj ↑	Gap	Time
Handcrafted Heuristic-based Classical Methods							
ILS C++	Heuristic [21]	5.98	0.00%	12h		–	
Compass	Heuristic [42]		–		33.19	0.00%	15m
RL-based Deep Constructive Heuristic methods with greedy rollout (zero-shot inference)							
AM $\{greedy.\}$	ICLR'19 [21]	6.25	4.46%	2s	31.62	4.75%	2s
MDAM $\{greedy. \times 5\}$	AAAI'21 [25]	6.17	3.13%	34s	32.32	2.61%	32s
Sym-NCO $\{greedy.\}$	*This work*	**6.05**	**1.23%**	2s	**32.51**	**2.03%**	2s
RL-based Deep Constructive Heuristic methods with multi-start rollout (Post-processing)							
AM {s. 1280}	ICLR'19 [21]	6.08	1.67%	27m	32.68	1.55%	25m
MDAM {bs. 30$\times$ 5}	AAAI'21 [25]	6.07	1.46%	16m	32.91	0.84%	14m
Sym-NCO {s. 200}	*This work*	**5.98**	**-0.02%**	3m	**33.04**	**0.45%**	3m

Average cost. We report an average cost of 10,000 benchmark instances which is proposed by [21].

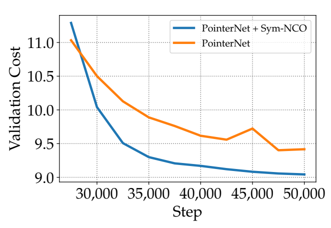
Evaluation speed. We report the evaluation speeds of solvers in a out-of-the-box manner as they are used in practice. In that regard, the execution time of non-neural and neural methods are measured on CPU and GPU, respectively.

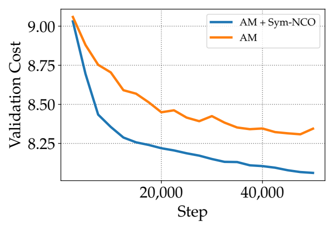
Greedy/Multi-start performance. For neural solvers, it is a common practice to measure *multi-start* performance as its final performance. However, when those are employed in practice, such resource consuming multi-start may not be possible. Hence, we discuss greedy and multi-start separately.

5.4 Experimental results

Results of TSP and CVRP. As shown in Table 1, Sym-NCO outperforms the NCO baselines in both the greedy rollout and multi-start settings with the fastest inference speed. Remarkably, Sym-NCO achieves a 0.95% gap in TSP using the greedy rollout. In the TSP greedy setting, it solves TSP 10,000 instances in a few seconds.

Results of PCTSP and OP. As shown in Table 2, Sym-NCO outperforms the NCO baselines in both the greedy rollout and multi-start settings. In the multi-start setting, Sym-NCO outperforms the classical PCTSP baseline (i.e., ILS) with the $\frac{43200}{180} \approx 240\times$ faster speed.


(a) PointerNet


(b) AM

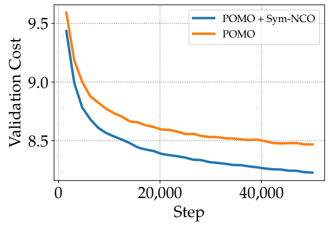
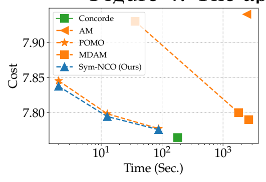
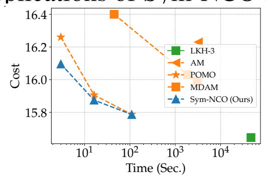
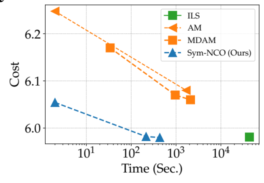

(c) POMO

Figure 4: The applications of Sym-NCO to DRL-NCO methods in TSP ($N = 100$)


(a) TSP ($N = 100$)


(b) CVRP ($N = 100$)


(c) PCTSP ($N = 100$)

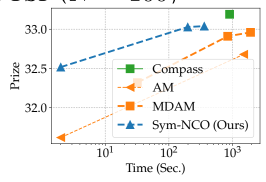

(d) OP ($N = 100$)

Figure 5: Time vs. cost plots. Green, orange, and blue colored lines visualize the results of hand-craft heuristics, neural baselines, and Sym-NCO, respectively. For OP (d), higher y-axis values are better.

Results of the real-world TSP. We evaluate POMO and Sym-NCO on TSPLib [43]. Table 3 shows that Sym-NCO outperforms POMO. Please refer to Appendix D.2 for the full benchmark results.

	Gap
POMO	1.87%
Sym-NCO	**1.62%**

Table 3: Optimality gap on TSPLIB

Application to various DRL-NCO methods. As discussed in Section 3, Sym-NCO can be applied to various various DRL-NCO methods. We validate that Sym-NCO significantly improves the existing DRL-NCO methods as shown in Fig. 4.

Time-performance analysis for multi-starts. Multi-starts is a common method that improves the solution qualities while requiring a longer time budget. We use the rotation augments [23] to produce multiple inputs (i.e., starts). As shown in Fig. 5, Sym-NCO achieves the Pareto frontier for all benchmark datasets. In other words, Sym-NCO exhibits the best solution quality among the baselines within the given time consumption.

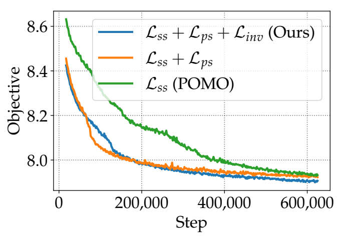
6 Discussion

6.1 Discussion of Regularization based Symmetricity Learning

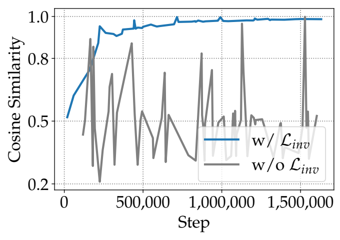
Ablation Study of $\mathcal{L}_{\text{inv}}$. As shown in Fig. 6b, $\mathcal{L}_{\text{inv}}$ increases the cosine similarity of the projected representation (i.e., $g(h)$). We can conclude that $\mathcal{L}_{inv}$ contributes to the performance improvements (see Fig. 6a). We further verify that imposing similarity on h degrades the performance as demonstrated in Fig. 6c. This proves the importance of maintaining the expression power of the encoder as we mentioned in Section 3.2.

Comparison with EGNN. EGNN [35] provably guarantees to process coordinate graph with rotational equivalency. Also, EGNN empirically verified its' high performance in point cloud tasks. Therefore, we implemented a simple EGNN-based CO policy (simply termed EGNN in this paper), to check the feasibility of CO. We leverage six EGNN layers with 128 hidden dimensions, to replace the POMO encoder where the POMO decoder is unchanged.

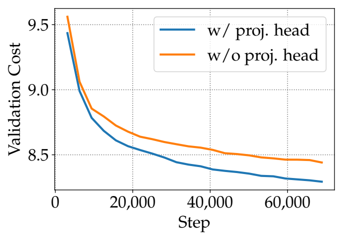
In the experimental results, we observed that EGNN significantly underperforms Sym-NCO and fails to converge as shown Fig. 7b. This is because the euclidian CO has a fully connected input graph containing informative coordinates, and we believe the equivariant neural network should be carefully crafted to consider such unique input graph structures of CO tasks. On the other hand, Sym-NCO could leverage the existing powerful NCO model without fine modification of the neural network. These numerical results conform well with our hypothesis that equivariance is necessary but not



(a) Cost curves of the loss designs

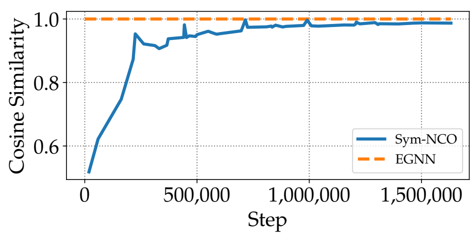


(b) Cosine similarity curves

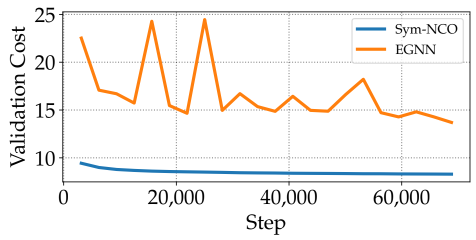


(c) Projection head ablation results

Figure 6: Loss design ablation results (a) Effect of loss components to the costs, (b) Cosine similarity curves of the models with and with $\mathcal{L}_{\text{inv}}$, (c) Costs of the models with and without $g(\cdot)$.



(a) Cosine similarity curves



(b) Cost curves

Figure 7: Comparisons of Sym-NCO and EGNN

sufficient for obtaining the optimal parameters for the solver. Therefore, we expect our approach can be simply extended to other domains requiring symmetricity and geometricity as positioned to leverage the existing legacy of powerful non-equivariant neural networks.

6.2 Limitations & future directions

Extended problem symmetricities. In this work, we employ the rotational symmetricity (Theorem 2.1) as the problem symmetricity. However, for some COPs, different problem symmetricities, such as scaling and translating P, can also be considered. Employing these additional symmetricities may further enhance the performance of Sym-NCO. We leave this for future research.

Large scale adaptation. Large scale applicability is essential to NCO. In this work we simply present scale adaptation capability of Sym-NCO using the effective active search (EAS) [44]; see Appendix D.3. We expect curriculum- [45], and meta-learning approaches may improve the generalizability of NCO to larger-sized problems.

Extension to the graph COP. This work finds the problem symmetricity that is universally applicable for *Euclidean* COPs. However, some COPs are defined in non-Euclidean spaces such as asymmetric TSP. Furthermore, there are also a bunch of existing neural combinatorial optimization models that can solve graph COP [46, 47, 48, 49, 50, 51], where we can improve with a symmetricity regularization scheme. We also leave finding the universal symmetricities of non-Euclidean COPs and applying them to existing graph COP models for future research.

6.3 Social Impacts

Design automation through NCO research affects various industries including logistics and transportation industries. From a negative perspective, this automation process may have some concerns to lead to unemployment in certain jobs. However, automation of logistics, transportation, and design automation can increase the efficiency of industries, reducing CO2 emissions (by reducing total tour length) and creating new industries and jobs.

Acknowledgments and Disclosure of Funding

We thank Jiwoo Son, Hyeonah Kim, Haeyeon Rachel Kim, Fangying Chen, and anonymous reviews for proving helpful feedback for preparing our manuscripts. This work was supported by a grant of the KAIST-KT joint research project through AI2XL Laboratory, Institute of convergence Technology, funded by KT [Project No. G01210696, Development of Multi-Agent Reinforcement Learning Algorithm for Efficient Operation of Complex Distributed Systems].

References

[1] Stefan Irnich, Paolo Toth, and Daniele Vigo. *Chapter 1: The Family of Vehicle Routing Problems*, pages 1–33.

[2] Matthew Veres and Medhat Moussa. Deep learning for intelligent transportation systems: A survey of emerging trends. *IEEE Transactions on Intelligent Transportation Systems*, 21(8):3152–3168, 2020.

[3] Sungsoo Ahn, Junsu Kim, Hankook Lee, and Jinwoo Shin. Guiding deep molecular optimization with genetic exploration. *Advances in neural information processing systems*, 33:12008–12021, 2020.

[4] Sungsoo Ahn, Binghong Chen, Tianzhe Wang, and Le Song. Spanning tree-based graph generation for molecules. In *International Conference on Learning Representations*, 2021.

[5] Azalia Mirhoseini, Hieu Pham, Quoc V. Le, Benoit Steiner, Rasmus Larsen, Yuefeng Zhou, Naveen Kumar, Mohammad Norouzi, Samy Bengio, and Jeff Dean. Device placement optimization with reinforcement learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2430–2439. PMLR, 06–11 Aug 2017.

[6] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Jiang, Ebrahim M. Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Sungmin Bae, Azade Nazi, Jiwoo Pak, Andy Tong, Kavya Srinivasa, William Hang, Emre Tuncer, Anand Babu, Quoc V. Le, James Laudon, Richard C. Ho, Roger Carpenter, and Jeff Dean. Chip placement with deep reinforcement learning. *CoRR*, abs/2004.10746, 2020.

[7] Haiguang Liao, Qingyi Dong, Xuliang Dong, Wentai Zhang, Wangyang Zhang, Weiyi Qi, Elias Fallon, and Levent Burak Kara. Attention routing: track-assignment detailed routing using attention-based reinforcement learning, 2020.

[8] Minsu Kim, Hyunwook Park, Seongguk Kim, Keeyoung Son, Subin Kim, Kyunjune Son, Seonguk Choi, Gapyeol Park, and Joungho Kim. Reinforcement learning-based auto-router considering signal integrity. In *2020 IEEE 29th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, pages 1–3, 2020.

[9] Minsu Kim, Hyunwook Park, Keeyoung Son, Seongguk Kim, Haeyeon Kim, Jihun Kim, Jinwook Song, Youngmin Ku, Jounggyu Park, and Joungho Kim. Imitation learning for simultaneous escape routing. In *2021 IEEE 30th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, pages 1–3, 2021.

[10] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28, pages 2692–2700. Curran Associates, Inc., 2015.

[11] Chaitanya K. Joshi, Quentin Cappart, Louis-Martin Rousseau, Thomas Laurent, and Xavier Bresson. Learning tsp requires rethinking generalization, 2020.

[12] Wouter Kool, Herke van Hoof, Joaquim A. S. Gromicho, and Max Welling. Deep policy dynamic programming for vehicle routing problems. *CoRR*, abs/2102.11756, 2021.

[13] Zhang-Hua Fu, Kai-Bin Qiu, and Hongyuan Zha. Generalize a small pre-trained model to arbitrarily large tsp instances, 2020.

[14] André Hottung, Bhanu Bhandari, and Kevin Tierney. Learning a latent search space for routing problems using variational autoencoders. In *International Conference on Learning Representations*, 2020.

[15] André Hottung and Kevin Tierney. Neural large neighborhood search for the capacitated vehicle routing problem. *CoRR*, abs/1911.09539, 2019.

[16] Yaoxin Wu, Wen Song, Zhiguang Cao, Jie Zhang, and Andrew Lim. Learning improvement heuristics for solving routing problems, 2020.

[17] Paulo R d O da Costa, Jason Rhuggenaath, Yingqian Zhang, and Alp Akcay. Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning. In Sinno Jialin Pan and Masashi Sugiyama, editors, *Proceedings of The 12th Asian Conference on Machine Learning*, volume 129 of *Proceedings of Machine Learning Research*, pages 465–480, Bangkok, Thailand, 18–20 Nov 2020. PMLR.

[18] Sungsoo Ahn, Younggyo Seo, and Jinwoo Shin. Learning what to defer for maximum independent sets. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 134–144. PMLR, 13–18 Jul 2020.

[19] Minsu Kim, Jinkyoo Park, and Joungho Kim. Learning collaborative policies to solve np-hard routing problems. In *Advances in Neural Information Processing Systems*, 2021.

[20] Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning, 2017.

[21] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019.

[22] Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takác. Reinforcement learning for solving the vehicle routing problem. *Advances in neural information processing systems*, 31, 2018.

[23] Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning. *Advances in Neural Information Processing Systems*, 33:21188–21198, 2020.

[24] Junyoung Park, Jaehyeong Chun, Sang Hun Kim, Youngkook Kim, and Jinkyoo Park. Learning to schedule job-shop problems: representation and policy learning using graph neural network and reinforcement learning. *International Journal of Production Research*, 59(11):3360–3377, 2021.

[25] Yining Ma, Jingwen Li, Zhiguang Cao, Wen Song, Le Zhang, Zhenghua Chen, and Jing Tang. Learning to iteratively solve routing problems with dual-aspect collaborative transformer. *Advances in Neural Information Processing Systems*, 34, 2021.

[26] Liang Xin, Wen Song, Zhiguang Cao, and Jie Zhang. Multi-decoder attention model with embedding glimpse for solving vehicle routing problems. In *Proceedings of 35th AAAI Conference on Artificial Intelligence*, pages 12042–12049, 2021.

[27] Junyoung Park, Sanjar Bakhtiyar, and Jinkyoo Park. Schedulenet: Learn to solve multi-agent scheduling problems with reinforcement learning. *arXiv preprint arXiv:2106.03051*, 2021.

[28] Xinyun Chen and Yuandong Tian. Learning to perform local rewriting for combinatorial optimization. In *Advances in Neural Information Processing Systems*, 2019.

[29] Hansen Wang, Zefang Zong, Tong Xia, Shuyu Luo, Meng Zheng, Depeng Jin, and Yong Li. Rewriting by generating: Learn heuristics for large-scale vehicle routing problems, 2021.

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

[31] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc., 2017.

[33] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics*, pages 4474–4484. PMLR, 2020.

[34] Fabian Fuchs, Daniel Worrall, Volker Fischer, and Max Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Advances in Neural Information Processing Systems*, 33:1970–1981, 2020.

[35] Vıéctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International Conference on Machine Learning*, pages 9323–9332. PMLR, 2021.

[36] Elise van der Pol, Daniel Worrall, Herke van Hoof, Frans Oliehoek, and Max Welling. Mdp homomorphic networks: Group symmetries in reinforcement learning. *Advances in Neural Information Processing Systems*, 33:4199–4210, 2020.

[37] Wenbin Ouyang, Yisen Wang, Paul Weng, and Shaochen Han. Generalization in deep rl for tsp problems via equivariance and local search. *arXiv preprint arXiv:2110.03595*, 2021.

[38] Benjamin Hudson, Qingbiao Li, Matthew Malencia, and Amanda Prorok. Graph neural network guided local search for the traveling salesperson problem. *arXiv preprint arXiv:2110.05291*, 2021.

[39] Vašek Chvátal David Applegate, Robert Bixby and William Cook. Concorde tsp solver.

[40] Keld Helsgaun. An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. 12 2017.

[41] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 6348–6358. Curran Associates, Inc., 2017.

[42] Gorka Kobeaga, María Merino, and Jose A Lozano. An efficient evolutionary algorithm for the orienteering problem. *Computers & Operations Research*, 90:42–59, 2018.

[43] Gerhard Reinelt. Tsplib—a traveling salesman problem library. *ORSA journal on computing*, 3(4):376–384, 1991.

[44] André Hottung, Yeong-Dae Kwon, and Kevin Tierney. Efficient active search for combinatorial optimization problems. *arXiv preprint arXiv:2106.05126*, 2021.

[45] Michal Lisicki, Arash Afkanpour, and Graham W Taylor. Evaluating curriculum learning strategies in neural combinatorial optimization. *arXiv preprint arXiv:2011.06188*, 2020.

[46] Thomas Barrett, William Clements, Jakob Foerster, and Alex Lvovsky. Exploratory combinatorial optimization with reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3243–3250, 2020.

[47] Iddo Drori, Anant Kharkar, William R Sickinger, Brandon Kates, Qiang Ma, Suwen Ge, Eden Dolev, Brenda Dietrich, David P Williamson, and Madeleine Udell. Learning to solve combinatorial optimization problems on real-world graphs in linear time. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 19–24. IEEE, 2020.

[48] Thomas D Barrett, Christopher WF Parsonson, and Alexandre Laterre. Learning to solve combinatorial graph partitioning problems via efficient exploration. *arXiv preprint arXiv:2205.14105*, 2022.

[49] Shenshen Gu and Yue Yang. A deep learning algorithm for the max-cut problem based on pointer network structure with supervised learning and reinforcement learning strategies. *Mathematics*, 8(2), 2020.

[50] Kenshin Abe, Zijian Xu, Issei Sato, and Masashi Sugiyama. Solving np-hard problems on graphs with extended alphago zero. *arXiv preprint arXiv:1905.11623*, 2019.

[51] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Combinatorial optimization with graph convolutional networks and guided tree search. *Advances in neural information processing systems*, 31, 2018.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Did you describe the limitations of your work? [Yes] See Section 6.2
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Source code will be available after decision is made.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 5.2.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix C.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Proof of Theorem 2.1

In this section, we prove the Theorem 2.1, which states a problem $\boldsymbol{P}$ and its' orthogonal transformed problem $Q(\boldsymbol{P}) = \{\{Qx_i\}_{i=1}^N, \boldsymbol{f}\}$ have identical optimal solutions if Q is orthogonal matrix: $QQ^T = Q^TQ = I$.

As we mentioned in Section 2.2, reward R is a function of $\boldsymbol{a}_{1:T}$ (solution sequences), $||x_i - x_j||_{i,j\in\{1,\ldots N\}}$ (relative distances) and $\boldsymbol{f}$ (nodes features).

For simple notation, let denote $||x_i - x_j||_{i,j\in\{1,\ldots N\}}$ as $||x_i - x_j||$. And Let $R^*(\boldsymbol{P})$ is optimal value of problem $\boldsymbol{P}$: i.e.

$$R^*(\boldsymbol{P}) = R(\boldsymbol{\pi}^*; \boldsymbol{P}) = R\left(\boldsymbol{\pi}^*; \{||x_i - x_j||, \boldsymbol{f}\}\right)$$

Where π^* is an optimal solution of problem $\boldsymbol{P}$. Then the optimal value of transformed problem $Q(\boldsymbol{P})$, $R^*(Q(\boldsymbol{P}))$ is invariant:

$$\begin{aligned}
R^*(Q(\boldsymbol{P})) &= R(\boldsymbol{\pi}^*; Q(\boldsymbol{P})) \\
&= R\left(\boldsymbol{\pi}^*; \{||Qx_i - Qx_j||, \boldsymbol{f}\}\right) \\
&= R\left(\boldsymbol{\pi}^*; \{\sqrt{(Qx_i - Qx_j)^T(Qx_i - Qx_j)}, \boldsymbol{f}\}\right) \\
&= R\left(\boldsymbol{\pi}^*; \{\sqrt{(x_i - x_j)^T Q^T Q(x_i - x_j)}, \boldsymbol{f}\}\right) \\
&= R\left(\boldsymbol{\pi}^*; \{\sqrt{(x_i - x_j)^T I(x_i - x_j)}, \boldsymbol{f}\}\right) \\
&= R\left(\boldsymbol{\pi}^*; \{||x_i - x_j||, \boldsymbol{f}\}\right) = R(\boldsymbol{\pi}^*; \boldsymbol{P}) = R^*(\boldsymbol{P})
\end{aligned}$$

Therefore, problem transformation of orthogonal matrix Q does not change the optimal value.

Then, the remaining proof is to show $Q(\boldsymbol{P})$ has an identical solution set with $\boldsymbol{P}$.

Let optimal solution set $\Pi^*(P) = \{\boldsymbol{\pi}^i(\boldsymbol{P})\}_{i=1}^M$, where $\boldsymbol{\pi}^i(\boldsymbol{P})$ indicates optimal solution of $\boldsymbol{P}$ and M is the number of heterogeneous optimal solution.

For any $\boldsymbol{\pi}^i(Q(\boldsymbol{P})) \in \Pi^*(Q(\boldsymbol{P}))$, they have same optimal value with $\boldsymbol{P}$:

$$R(\boldsymbol{\pi}^i(Q(\boldsymbol{P})); Q(\boldsymbol{P})) = R^*(Q(\boldsymbol{P})) = R^*(\boldsymbol{P})$$

Thus, $\boldsymbol{\pi}^i(Q(\boldsymbol{P})) \in \Pi^*(P)$.

Conversely, For any $\boldsymbol{\pi}^i(\boldsymbol{P}) \in \Pi^*(\boldsymbol{P})$, they have sample optimal value with $Q(\boldsymbol{P})$:

$$R(\boldsymbol{\pi}^i(\boldsymbol{P}); \boldsymbol{P}) = R^*(\boldsymbol{P}) = R^*(Q(\boldsymbol{P}))$$

Thus, $\boldsymbol{\pi}^i(\boldsymbol{P}) \in \Pi^*(Q(\boldsymbol{P}))$.

Therefore, $\Pi^*(\boldsymbol{P}) = \Pi^*(Q(\boldsymbol{P}))$, i.e., $\boldsymbol{P} \xleftrightarrow{\text{sym}} Q(\boldsymbol{P})$.

B Implementation of Baselines

We directly reproduce competitive DRL-NCO methods: POMO [23] and AM [21] and PointerNet [10, 20].

PointerNet. The PointerNet is early work of DRL-NCO using LSTM-based encoder-decoder architecture trained with actor-critic manner. We follow the instruction of open source code [1] by [21] following hyperparmeters.

REINFORCE baseline	Rollout baseline [21]
Learning rate	1e-4
The Number of Encoder Layer	3
Embedding Dimension	128
Batch-size	512
Epochs	100
Epoch size	1,280,000
The Number of Steps	250K

Table 4: Hyperparameter Setting for AM for all tasks.

AM. The AM is a general-purpose DRL-NCO, a transformer-based encoder-decoder model that solves various routing problems such as TSP, CVRP, PCTSP, and OP. We follow the instruction of open source code, same with the PointerNet with the following hyperparameters.

REINFORCE baseline	Rollout baseline [21]
Learning rate	1e-4
The Number of Encoder Layer	3
Embedding Dimension	128
Attention Head Number	8
Feed Forward Dimension	512
Batch-size	512
Epochs	100
Epoch size	1,280,000
The Number of Steps	250K

Table 5: Hyperparameter Setting for AM for all tasks.

POMO. The POMO is a high-performance DRL-NCO for TSP and CVRP, implemented on the top of the AM. We follow the instruction of open source code [2] with the following hyperparameters.

	TSP	CVRP
REINFORCE baseline	POMO shared baseline [23]	
Learning rate	1e-4	
Weight decay	1e-6	
The Number of Encoder Layer	6	
Embedding Dimension	128	
Attention Head Number	8	
Feed Forward Dimension	512	
Batch-size	64	
Epochs	2,000	8,000
Epoch size	100,000	10,000
The Number of Steps	3.125M	1.25M

Table 6: Hyperparameter Setting for POMO in TSP and CVRP.

[1]https://github.com/wouterkool/attention-learn-to-route
[2]https://github.com/yd-kwon/POMO

C Implementation Details of Proposed Method

C.1 Training Hyperparameters

Sym-NCO is a training scheme that is attached to the top of the existing DRL-NCO model. We set the same hyperparameters with PointerNet, AM, and POMO Appendix B except REINFORCE baseline (we set the proposed Sym-NCO baseline introduced in Section 3).

Sym-NCO has additional hyperparameters. First of all, we set identical hyperparameters for PointerNet and AM for all tasks:

α	0.1
β	0
K	1
L	10

Table 7: Hyperparameter Setting of Sym-NCO for PointerNet and AM.

Note that the design choice of $\beta = 0$ is to show high applicability of $\mathcal{L}_{\text{ps}}$, and is because AM with $\mathcal{L}_{\text{ss}}$ is just similar to the POMO.

For POMO, we set $\beta = 1$ to force solution symmetricity on the top of POMO's baseline. Note that we follow POMO's first node restriction only in TSP, which is a reasonable bias as we mentioned in Section 3. The hyperparameter setting is as follows:

	TSP	CVRP
α	0.1	0.2
β	1	1
K	100	100
L	2	2

Table 8: Hyperparameter Setting of Sym-NCO for POMO.

Note that the design choice of $\beta = 1$ and $K = 100$ is based on POMO's baseline setting. We just set $L = 2$, because of training efficiency. We suggest to set $L > 4$ if training resources and time-budget is sufficient;; it may increase performance further.

C.2 Integration of $\mathcal{L}_{\text{ss}}$ and $\mathcal{L}_{\text{ps}}$

The $\mathcal{L}_{\text{ps}}$ is an extension of $\mathcal{L}_{\text{ss}}$ where it can both leverage problem symmetricity and solution symmetricity. Therefore, we can simply use $\mathcal{L}_{\text{RL-Sym}} = \mathcal{L}_{\text{ps}}$. However, some specific CO problem such as TSP has cyclic nature, which contains pre-identifiable solution symmetricity, and some method already exploit the cyclic nature. For example, the POMO [23] which is a powerful NCO model already utilizes pre-identified solution symmetricity in the training process for specific CO tasks. Therefore, we provide a general loss term $\mathcal{L}_{\text{RL-Sym}} = \mathcal{L}_{\text{ps}} + \beta\mathcal{L}_{\text{ss}}$ that can be used with POMO or similar methods for specific CO problems (TSP and CVRP). If the problem has pre-identified solution symmetricity (TSP) or has a strong cyclic nature (CVRP), we can set $\beta = 1$ to leverage solution-symmetricity more. If we do not have specific domain knowledge for the target task, then we leave $\beta = 0$, to leverage problem-symmetricity and solution-symmetricity simultaneously using only $\mathcal{L}_{\text{ps}}$.

C.3 Multi-start Post-processing

To sample multi solutions from one solver F_θ we suggest instance augmentation method following [23]. As suggested in [23], we can generate multiple samples to ablate first node selection of decoding step by N. Moreover we can generate 8 samples to rotate with $0, 90, 180, 270$ degrees with reflection: $4 \times 2 = 8$. To comparison with Sym-NCO and POMO as shwon in Fig. 5 (three markers), we conduct these multi state post processing with sampling width: $1, 100, 100 \times 8$.

We, on the other hand, suggest an extended version of the instance augmentation method of [23], using random orthogonal matrix Q. By transforming input problem $\boldsymbol{P}$ with $Q^1, ..., Q^M$ which are orthogonal matrices, we can sample multiple sample solutions from the M symmetric problems. We used these strategies in PCTSP and OP by setting $M = 200$.

C.4 Details of Projection Head

The projection head introduced in Section 3.2 is a simple two-layer perception with the ReLU activation function, where input/output/hidden dimensions are equals to encoder's embedding dimension (i.e. 128).

C.5 Computing Resources and Computing Time

For training Sym-NCO, we use *NVIDIA* A100 GPU. Because POMO implementation does not support GPU parallelization, we use a single GPU for the POMO + Sym-NCO. It takes approximately two weeks to finish training POMO + Sym-NCO. For training AM + Sym-NCO, we use $4\times$ GPU, which takes approximately three days to finish training.

As mentioned in Section 5.2, we use *NVIDIA* RTX2080Ti single GPU at the test time.

D Additional Experiments

D.1 Hyperparameter Tuning of α in CVRP

We did not tune hyperparameter much in this work because training resources were limited where Sym-NCO must be verified on several tasks and DRL-NCO architectures. Therefore, we only contain simple hyperparameter ablation for α in CVRP (POMO + SymNCO setting).

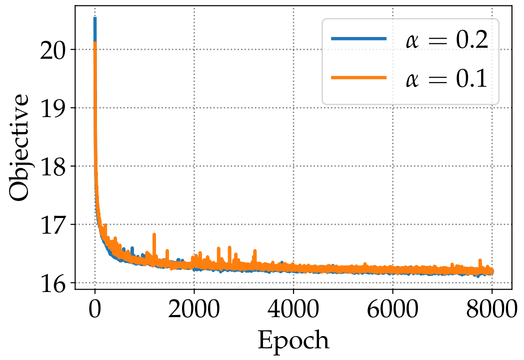


Figure 8: Alblation Study for $\alpha \in \{0.1, 0.2\}$

This validation results shows $\alpha = 0.2$ give slightly better performances than $\alpha = 0.1$, but tuning of α semms to be not sensitive.

D.2 Performance Evaluation on TSPLIB

This section gives Sym-NCO performance evaluation in the TSPLIB ($N < 250$). Sym-NCO and the POMO is pre-trained model in $N = 100$ that is evaluated in Table 1. In this experiment, we conduct multi-start sampling with sample width $M = N \times 20$ where the N indicates multi initial city sampling of problem size (ex., the “eil51" has $N = 51$). The 20 indicates multi-sampling using random orthogonal matrix as we introduced in Appendix C.5. As shown in the below table, our Sym-NCO outperforms POMO, having a 1.62% optimal gap, which is extremely high performance in real-world TSPLIB evaluation compared with other NCO evaluations [19].

Table 9: Performance comparison in real-world instances in TSPLIB.

Instance	Opt.	POMO [23]		Sym-NCO (ours)	
		Cost	Gap	Cost	Gap
eil51	426	429	0.82%	432	1.39%
berlin52	7,542	7,545	0.04%	7,544	0.03%
st70	675	677	0.31%	677	0.31%
pr76	108,159	108,681	0.48%	108,388	0.21%
eil76	538	544	1.18%	544	1.18%
rat99	1,211	1,270	4.90%	1,261	4.17%
rd100	7,910	7,912	0.03%	7,911	0.02%
KroA100	21,282	21,486	0.96%	21,397	0.54%
KroB100	22,141	22,285	0.65%	22,378	1.07%
KroC100	20,749	20,755	0.03%	20,930	0.87%
KroD100	21,294	21,488	0.91%	21,696	1.89%
KroE100	22,068	22,196	0.58%	22,313	1.11%
eil101	629	641	1.84%	641	1.84%
lin105	14,379	14,690	2.16%	14,358	0.54%
pr124	59,030	59,353	0.55%	59,202	0.29%
bier127	118,282	125,331	5.96%	122,664	3.70%
ch130	6,110	6,112	0.03%	6,118	0.14%
pr136	96,772	97,481	0.73%	97,579	0.83%
pr144	58,537	59,197	1.13%	58,930	0.67%
kroA150	26,524	26,833	1.16%	26,865	1.28%
kroB150	26,130	26,596	1.78%	26,648	1.98%
pr152	73,682	74,372	0.94%	75,292	2.18%
u159	42,080	42.567	1.16%	42,602	1.24%
rat195	2,323	2,546	9.58%	2,502	7.70%
kroA200	29,368	29,937	1.94%	29,816	1.53%
ts225	126,643	131,811	4.08%	127,742	0.87%
tsp225	3,919	4,149	5.87%	4,126	5.27%
pr226	80,369	82,428	2.56%	82,337	2.45%
Avg Gap	0.00%	1.87%		**1.62**%	

D.3 Performance Evaluation of Transferability to Large Scale Problems

This section verifies that the pre-trained model using the Sym-NCO has powerful transferability on large-scale problems. We use the efficient-active-search (EAS) [44] as a transfer learning algorithm for large-scale TSP [3]. In transfer learning, the number of iterations is an important factor for adaptation. We set the iteration $K = 200$ as default; we provide an ablation study for few-shot learning $K \in \{1, 2, 5, 10\}$ to show Sym-NCO's few-shot adaptation capability. Note that the pre-trained model is trained on CVRP ($N = 100$).

As shown in Table 10, our method outperforms the POMO [23] in large-scale CVRP, having only a small performance gap with LKH3. Furthermore, our method increase few shot adaptation capabilities for large-scale tasks, where our model achieved better performances than POMO with 5 $\times$ reduced training shot K. To sum up, our Sym-NCO can be positioned with an effective pretraining scheme that approximately imposes symmetricity and is further transferred to larger-scale tasks.

Table 10: Performance comparison in large scale CVRP. The performance is evaluated on ten random generated CVRP data.

	CVRP ($N = 500$)		CVRP ($N = 1,000$)	
	Cost	Gap	Cost	Gap
LKH3 [40]	60.37	0.00%	115.74	0.00%
POMO [23] + EAS [44]	63.30	4.85%	126.56	9.34%
Ours + EAS [44]	**62.41**	**3.37%**	**121.85**	**5.92%**

Table 11: Performance evaluation of few shot adaptation to large scale CVRP.

	CVRP ($N = 500$)				CVRP ($N = 1,000$)			
	$K = 1$	$K = 2$	$K = 5$	$K = 10$	$K = 1$	$K = 2$	$K = 5$	$K = 10$
POMO [23] + EAS [44]	136.91	116.77	77.57	69.90	366.61	311.41	189.26	162.64
Sym-NCO + EAS [44]	**75.85**	**69.71**	**67.26**	**66.33**	**192.12**	**163.92**	**139.66**	**134.61**

[3] All the hyperparameters are the same with https://github.com/ahottung/EAS

D.4 Comparison with Deep Improvement Heuristic Methods

In this section, we provide performance comparison with state-of-the-art deep improvement heuristics. As shown in Table 12, our method outperformed state-of-the-art deep improvement heuristics with the fastest speed. Note that constructive heuristics (which include our method) and improvement heuristics are complementary and can support each other.

Table 12: Performance comparison with deep improvement heuristics. The I indicates the number of iterations, and the s indicates the number of samples per instance.

	TSP ($N = 100$)		TSP ($N = 100$)	
	Cost	Gap	Cost	Gap
Wu et al. ($I = 5K$) [16] (I=5K)	1.42%	2h	2.47%	5h
DACT ($I = 1K$) [25]	1.62%	48s	3.18%	2m
DACT ($I = 5K$) [25]	0.61%	4m	1.55%	8m
Ours (s.100)	**0.39**%	12s	**1.46**%	16s
Ours (s.800)	**0.14**%	1m	**0.90**%	2m

We remark that the speed evaluation of Wu et al. [16] and DACT [25] is referred to by [25] where the speed is evaluated with NVIDIA TITAN RTX. The speed of our method is evaluated with NVIDIA RTX 2080Ti.

D.5 Comparison with Symmetric NCO models

Previous Symmetricitcy Considered NCO methods vs. Sym-NCO. Several studies exploited the symmetric nature of CO. Ouyang et al. [37] have a similar purpose to Sym-NCO in that both are DRL-based constructive heuristics, but they give rule-based input transformation (relative position from first visited city) to satisfy equivariance. However, our method learns to impose symmetricity approximately into the neural network with regularization loss term. We believe our approach is a more general approach to tackling symmetricity (see Table 14) because not every task can be represented as a relative position with the first visited city.

The Hudson et al. [38] is the extended work of Joshi et al. [11] where graph neural network (GNN) makes a sparse graph from a fully connected input graph, and the search method figures out the feasible solution from the sparse graph. This method is based on the supervised learning scheme that requires expert labels. Moreover, this method does not guarantee to generate feasible solutions in hard-constraint CO tasks because the pruning process of the GNN may eliminate feasible trajectory (In TSP, it may work, but in other tasks, this method must address feasibility issues). Regardless of this limitation, we view the line graph transformation suggested by Hudson et al. [38] as novel and helpful in terms of symmetricity.

Ma et al. [25] proposed a DRL-based improvement heuristic, exploiting the cyclic nature of TSP and CVRP. The purpose of Ma et al.[25], and our Sym-NCO is different: the objective of Sym-NCO is approximately imposing symmetricity nature, but the objective of Ma et al. [25] is to improve the iteration process of improvement heuristic with fined designed positional encoding for TSP and CVRP. Note that Sym-NCO (constructive method) and Ma et al. [25] (Improvement method) are complementary and can support each other. For example, pretrained constructive model can generate an initial high-quality solution, whereas an improvement method can iteratively improve solution quality.

Experimental Comparison.

Our Sym-NCO outperforms all the relevant related baselines as shown in Table 13. Furthermore, Table 14 shows our method covers the widest arrange of CO tasks, where it does not needs labeled data.

Table 13: Performance comparison with symmetric NCO methods

	Optimal Gap	Time	GPU resources
Ouyang et al. [37]	2.61%	1.3m	GTX1080Ti
Hudson et al. [38]	0.698%	28h	Tesla P100
Ma et al. [25]	1.62%	48s	Titan RTX
Ours	**0.39**%	**12s**	RTX 2080Ti

Table 14: Performance comparison with symmetric NCO methods

	Learning Methods	Verified Tasks
Ouyang et al. [37]	Reinforcement Learning	TSP
Hudson et al. [38]	Supervised Learning	TSP
Ma et al. [25]	Reinforcement Learning	TSP, CVRP
Ours	Reinforcement Learning	TSP, CVRP, PCTSP, OP