Sym-NCO: Leveraging Symmetricity for Neural Combinatorial Optimization

Anonymous Author(s)
Affiliation
Address
email

Abstract

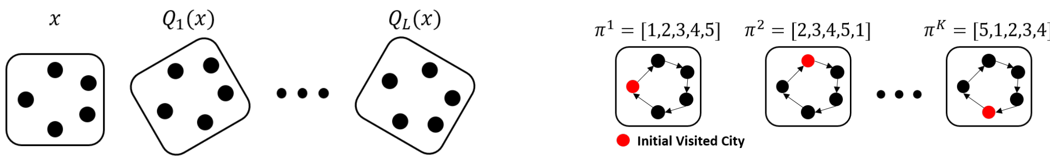
Deep reinforcement learning (DRL)-based combinatorial optimization (CO) methods (i.e., DRL-NCO) have shown significant merit over the conventional CO solvers as DRL-NCO is capable of learning CO solvers without supervised labels attained from the verified solver. This paper presents a novel training scheme, Sym-NCO, that achieves significant performance increments to existing DRL-NCO methods. Sym-NCO is a regularizer-based training scheme that leverages universal symmetricities in various CO problems and solutions. Imposing symmetricities such as rotational and reflectional invariance can greatly improve the generalization capability of DRL-NCO as symmetricities are invariant features shared by various CO tasks. Our experimental results verify that Sym-NCO greatly improved the performance of DRL-NCO methods in four CO tasks, including traveling salesman problem (TSP), capacitated vehicle routing problem (CVRP), prize collecting TSP (PCTSP), and orienteering problem (OP), without employing problem-specific techniques. 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× faster speed. Source code will be available after the decision is made.

1 Introduction

Combinatorial optimization problems (COPs) are mathematical optimization problems on discrete input space that carry numerous valuable applications, including vehicle routing problems (VRPs) [1, 2], drug discovery [3, 4], and semi-conductor chip design [5, 6, 7]. 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 were solved by integer program (IP) solvers or hand-crafted (meta) heuristics. Recent advances in computing infrastructures and deep learning 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 [8, 9, 10, 11, 12] and reinforcement learning (RL) [13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25]. Depending on the solution generation scheme, NCO methods are also classified into improvement [15, 14, 13, 26, 16, 17, 23] and constructive heuristics [18, 19, 20, 21, 22, 24, 25]. Among the NCO approaches, deep RL (DRL)-based constructive heuristics (i.e., DRL-NCO) are favored over conventional approaches due to the train-ability of RL that does not rely on existing COP solvers, and the tractability of the

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.



(a) Problem symmetricity (b) Solution symmetricity

Figure 1: Illustration of symmetricities in CO (exampled in TSP)

constructive process that prevents rule-violation of specific task and guarantees qualified solutions [19].

Despite the strength of DRL-NCO, there exists a performance gap between the state-of-the-art 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 [21, 27]. 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. To this end, we first identify the symmetricities present in various COPs. Sym-NCO leverages two types of symmetricities innate in COP that are defined on the Euclidean graph. First, the problem symmetricity derived from rotational invariance of the solution; the rotated graph must exhibit the same optimal solution as the original graph as shown in Fig. 1a. Second, the solution symmetricity, which is the shared feature among solutions having identical optimal values. For example, the solution symmetricity in the traveling salesmen problem (TSP) includes the first-city permutation invariance (See Fig. 1b). However, the solution symmetricity of general COPs must be automatically identified during the training process. That is because the shared feature between multiple optimal solutions is usually intractable without highly investigated domain knowledge.

The Sym-NCO is composed of two novel regularization methods for leveraging symmetricities. First, we suggest a new advantage function on REINFORCE algorithm that automatically identifies and exploits symmetricities without imposing misleading bias. Second, we devise a novel representation learning scheme to impose symmetricities by leveraging the pre-identified symmetricities.

We experimentally validated Sym-NCO on various existing DRL-NCO methods by solving their original target problems without employing any problem-specific techniques. By leveraging the symmetricities of COPs, Sym-NCO achieved the following:

- **High performances.** Sym-NCO achieved near-optimal performance in various COP tasks (less than 2%) with extremely high speed (few seconds to solve 10,000 instances). Moreover, Sym-NCO surpassed the competitive PCTSP solver, ILS [19], at 240× faster speed.
- **Problem agnosticism.** Sym-NCO does not employ problem-specific heuristics to solve various COPs. Sym-NCO is generally applicable to solve TSP, CVRP PCTSP, and OP.
- **Architecture agnosticism.** Sym-NCO can easily be implemented to any encoder-decoder model and impose the symmetricities of COPs. Sym-NCO successfully improved the performance of existing encoder-decoder-based DRL-NCO methods, such as PointerNet [8, 18], AM [19] and POMO [21].

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:

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

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, $\boldsymbol{\pi}(\boldsymbol{P})$ is a solution of $\boldsymbol{P}$, and $R(\boldsymbol{\pi}(\boldsymbol{P}))$ is the objective value of $\boldsymbol{\pi}(\boldsymbol{P})$.

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 $R(\boldsymbol{\pi}(P))$ is the objective of COP. 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:

$$\boldsymbol{\pi}(P) \sim F_\theta(P) = \prod_{t=1}^{T} p_\theta(a_t|\boldsymbol{s}_t(P)) \tag{2}$$

where $p_\theta(a_t|\boldsymbol{s}_t(\boldsymbol{P}))$ 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 ($\boldsymbol{P}^i \xleftrightarrow{\text{sym}} \boldsymbol{P}^j$) if their optimal solution sets are identical.

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

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., $\boldsymbol{P} \xleftrightarrow{\text{sym}} Q(\boldsymbol{P})$. 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 imposes the symmetricities on F_θ by minimizing the symmetric loss function that is defined as follows:

$$\mathcal{L}_{\text{sym}} = \alpha\mathcal{L}_{\text{inv}} + \beta\mathcal{L}_{\text{ss}} + \mathcal{L}_{\text{ps}} \tag{3}$$

where $\mathcal{L}_{\text{inv}}$, $\mathcal{L}_{\text{ps}}$, and $\mathcal{L}_{\text{ss}}$ are the loss functions that impose invariant representation, problem-solution joint symmetricity, and solution symmetricity, respectively. $\alpha, \beta \in [0, 1]$ are the weight coefficients. In the following subsections, we explain each loss component in detail.

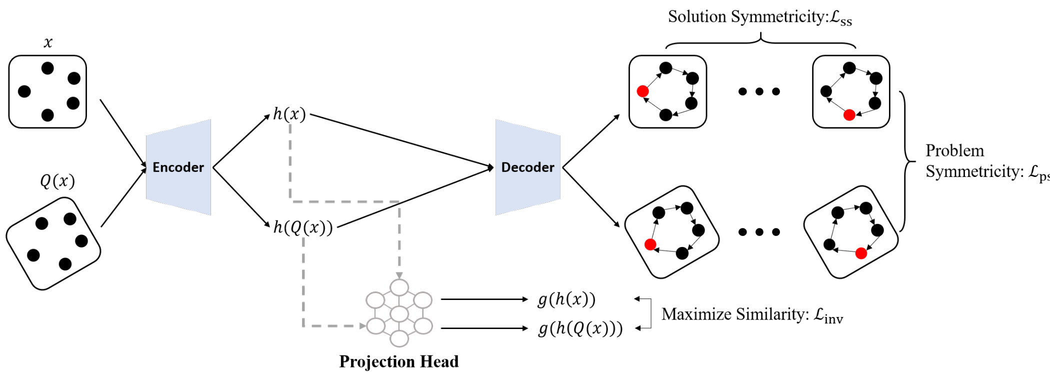


Figure 2: An overview of Sym-NCO

3.1 Imposing invariant problem representation via $\mathcal{L}_{\text{inv}}$.

By Theorem 2.1, the original problem x and its rotated problem $Q(x)$ have identical solutions. We impose this solution symmetricity on the encoder of F_θ by using the rotational invariant representation.

We denote $h(x)$ and $h\left(Q(x)\right)$ as the hidden representations of x and $P(x)$, respectively. To impose the rotational invariant property on $h(x)$, we define $\mathcal{L}_{\text{inv}}$ as follows:

$$\mathcal{L}_{\text{inv}} = -S_{\cos}\Big(g\Big(h(x)\Big), g\Big(h(Q(x))\Big)\Big) \tag{4}$$

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

To impose the rotational invariance, 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 [28]. We empirically verified that this approach attains stronger solvers as described in Section 6.1.

Note that the rotational invariance of h can also be attained through the EGNN [29], an invariant encoder architecture. However, this approach inevitably restricts the flexibility of the encoder architecture, thus preventing the use of flexible architectures, like Transformer [30]. We have empirically validated that our approach is more effective than the EGNN.

3.2 Imposing problem and solution symmetricities via $\mathcal{L}_{\text{ps}}$ and $\mathcal{L}_{\text{ss}}$

As discussed in Section 2.2, COPs have problem and solution symmetricities. We explain how to impose the symmetricities by minimizing $\mathcal{L}_{\text{ps}}$ and $\mathcal{L}_{\text{ss}}$ when training F_θ. We provide the policy gradients to $\mathcal{L}_{\text{ss}}\left(\boldsymbol{\pi}(\boldsymbol{P})\right)$ and $\mathcal{L}_{\text{ps}}\left(\boldsymbol{\pi}(\boldsymbol{P})\right)$ in the context of the REINFORCE algorithm [31] with the proposed baseline scheme.

Imposing solution symmetricity. In general COPs, symmetric solutions are usually intractable. As defined in Definition 2.2, the symmetric solutions must have the same objective values. Hence, we regularize $F_\theta(\boldsymbol{\pi}|\boldsymbol{P})$ with the gradient of $\mathcal{L}_{\text{ss}}(\boldsymbol{\pi}(P))$ so that its realized solutions have the same objective value. The gradient of $\mathcal{L}_{\text{ss}}(\boldsymbol{\pi}(P))$ is as follows:

$$\nabla_\theta \mathcal{L}_{\text{ss}}(\boldsymbol{\pi}(\boldsymbol{P})) = -\mathbb{E}_{\boldsymbol{\pi}^k \sim F_\theta(\cdot|\boldsymbol{P})}\Big[\underbrace{\Big[R(\boldsymbol{\pi}(\boldsymbol{P})) - \overbrace{\frac{1}{K}\sum_{k=1}^{K} R(\boldsymbol{\pi}^k)}^{Baseline}\Big]}_{Advantage} \nabla_\theta \log F_\theta\big(\boldsymbol{\pi}(\boldsymbol{P})\big)\Big] \tag{5}$$

where $\{\pi^k\}_{k=1}^K$ are the solutions of P sampled from $F_\theta(\pi|\boldsymbol{P})$, and $\log F_\theta(\pi(\boldsymbol{P}))$ is the log-likelihood of F_θ to generate $\pi(\boldsymbol{P})$. K is the number of sampled solutions.

The REINFORCE trains a solver by maximizing the expected reward, which is saturated in the end (i.e., sub-optimal solutions can be found) if properly trained. Saturation of reward value indicates a small deviation of rewards between sampled solutions $\pi^1, ..., \pi^K$, which refers to the (near) solution symmetricity as defined in Definition 2.2. Although naive REINFORCE has the ability to impose solution symmetricity by nature, it does not guarantee that the identified symmetric solutions are optimal. Our proposed baseline guides the solver to find improved symmetric solutions by inducing competition within the symmetric solution groups. If any of a symmetric solution group has the higher optimality over the rest of solutions in the group, the proposed advantage of the rest of solutions becomes a negative value. Negative advantage then encourages the solver to find the better solution so that prevents to fall in a bad local optimum. Overall, the proposed $\mathcal{L}_{\text{ss}}$ not only imposes solution symmetricity but also guides to find near-optimal solutions.

POMO [21] employs a similar training technique that finds symmetric solutions by forcing F_θ to visit all possible initial cities when solving TSP and CVRP. For TSP, the training technique is justifiable as the permutation of the initial cities preserves the reward. However, the reward of COPs, including CVRP, PCTSP, and OP, is usually sensitive to first city selection. To this end, we do not restrict first city selection (except TSP) but leave it to the training process of $\mathcal{L}_{\text{ss}}$. Therefore, the solution symmetricity must be identified through the training process without employing potentially misleading bias. Though the solution symmetricity is a significant feature to increase generalization capability, it is not always present nor found in all COPs. Such observations highlight the limitations of leveraging only the solution symmetricities when deriving F_θ. This motivates us to devise a more general method to leverage symmetricities in COPs.

Imposing problem and solution symmetricities. As discussed in Section 2.2, the rotational problem symmetricity is common in various COPs. Thus, we regularize F_θ in terms of the rotational problem symmetricity with the gradient of $\mathcal{L}_{\text{ps}}(\pi(\boldsymbol{P}))$ defined as follows:

$$\nabla_\theta \mathcal{L}_{\text{ps}}(\pi(\boldsymbol{P})) = -\mathbb{E}_{Q^l \sim \mathbf{Q}}\left[\mathbb{E}_{\pi^{l,k} \sim F_\theta(\cdot|Q^l(\boldsymbol{P}))}\left[\underbrace{\left[R(\pi(\boldsymbol{P})) - \overbrace{\frac{1}{LK}\sum_{l=1}^{L}\sum_{k=1}^{K} R(\pi^{l,k})}^{Baseline}\right]}_{Advantage} \nabla_\theta \log F_\theta(\pi(\boldsymbol{P}))\right]\right] \tag{6}$$

where $\mathbf{Q}$ is the distribution of random orthogonal matrices, Q^l is the l^{th} sampled rotation matrix, and $\pi^{l,k}$ is the k^{th} sample solution of the l^{th} rotated problem. L and K are the number of the sampled rotation matrix and solution symmetric solutions, respectively.

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 as x and $Q_l(x)$ having 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; $\mathcal{L}_{\text{ps}}$ are applicable to general COPs. Moreover, multiple solutions are sampled for each symmetric problem so that $\mathcal{L}_{\text{ps}}$ can impose solution symmetricity with a similar approach taken for $\mathcal{L}_{\text{ss}}$.

4 Related Works

Deep construction heuristics Bello et al. [18] propose one of the earliest DRL-NCO methods, based on PointerNet [8], and trained it with an actor-critic method. Attention model (AM) [19] successfully extends [18] by swapping PointerNet with Transformer [30], 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, 17]. POMO [21] 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 (i.e., not problem agnostic). MDAM [24] extends AM by employing an ensemble of decoders. However, such extension is inapplicable for stochastic routing problems (i.e., not problem agnostic). Among these promising DRL-NCO methods, Sym-NCO achieved SOTA performances with *problem-agnostic* properties.

Equivariant deep learning In deep learning, symmetricities are often enforced by employing specific network architectures. EDP-GNN [32] proposes a permutation equivariant graph neural network (GNN) that produces equivariant outputs to the input order permutations. $SE(3)$-Transformer [33] restricts the Transformer so that it is equivariant to $SE(3)$ group input transformation. Similarly, EGNN [29] 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 [34]. However, imposing the symmetricities via specialized network architecture (i.e., *hardly* constrained to satisfy the symmetries) can limit the representation capabilities of the models, which consequently limits the solution quality of DRL-NCO. We further discuss this issue in Section 6.1.

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 [21] to solve TSP and CVRP, and AM [19] to solve PCTSP and OP, respectively. We additionally validate the effectiveness of Sym-NCO on PointerNet [8].

5.1 Tasks and baseline selections

TSP aims to find the Hamiltonian cycle with a minimum tour length. We employ Concorde [35] and LKH-3 [36] as the non-learnable baselines, and PointerNet [8], S2V-DQN [37], RL [20] AM [19], POMO [21] and MDAM [24] 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 [36] as the non-learnable baselines, and RL[20], AM [19], POMO [21], and MDAM [24] 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) [19] as the non-learnable baseline, and AM [19] and MDAM [24] 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* [38] as the non-learnable baseline, and AM [19] and MDAM [24] 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 [39].

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 [19] 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 [24].

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 [35]	7.76	0.00%	3m		–	
LKH3	Heuristic [36]	7.76	0.00%	21m	15.65	0.00%	13h
RL-based Deep Constructive Heuristic methods with greedy rollout							
PointerNet $\{gr.\}$	NIPS'14 [8, 18]	8.30	6.90 %	–		–	
S2V-DQN $\{gr.\}$	NeurIPS'17 [37]	8.31	7.03 %	–		–	
RL $\{gr.\}$	NeurIPS'18 [20]		–		17.23	10.12%	–
AM $\{gr.\}$	ICLR'19 [19]	8.12	4.53%	2s	16.80	7.34%	3s
POMO $\{gr.\}$	NeurIPS'20 [21]	7.85	1.04%	2s	16.26	3.93%	3s
MDAM {gr.$\times$ 5}	AAAI'21 [23]	7.93	2.19%	36s	16.40	4.86%	45s
Sym-NCO $\{gr.\}$	*This work*	**7.84**	**0.94**%	2s	**16.10**	**2.88**%	3s
RL-based Deep Constructive Heuristic methods with multi-start rollout							
RL $\{bs.10\}$	NeurIPS'18 [20]		–		16.96	8.39%	–
AM $\{s.1280\}$	ICLR'19 [19]	7.94	2.26%	41m	16.23	3.72%	54m
POMO {s. 100}	NeurIPS'20 [21]	7.80	0.44%	13s	15.90	1.67%	16s
MDAM {bs. 30 $\times$ 5}	AAAI'21 [23]	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 [19]	5.98	0.00%	12h		–	
Compass	Heuristic [38]		–		33.19	0.00%	15m
RL-based Deep Constructive Heuristic methods with greedy rollout (zero-shot inference)							
AM {gr.}	ICLR'19 [19]	6.25	4.46%	2s	31.62	4.75%	2s
MDAM {gr.$\times$ 5}	AAAI'21 [23]	6.17	3.13%	34s	32.32	2.61%	32s
Sym-NCO {gr.}	*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 [19]	6.08	1.67%	27m	32.68	1.55%	25m
MDAM {bs. 30$\times$ 5}	AAAI'21 [23]	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

5.3 Performance metrics

This section provides detailed performance metrics:

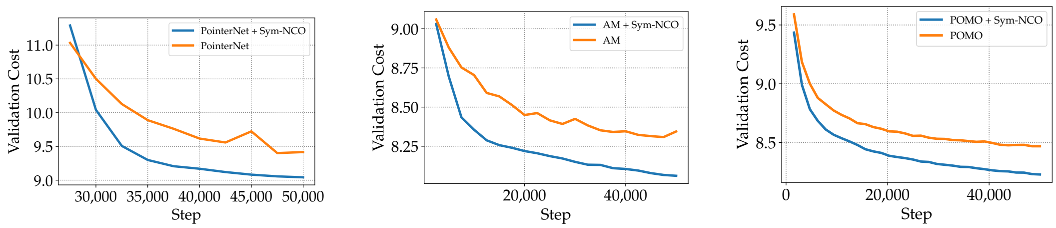
Average cost. We report an average cost of 10,000 benchmark instances which is proposed by [19].

Evaluation speed. We report the evaluation speeds of solvers in an 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.

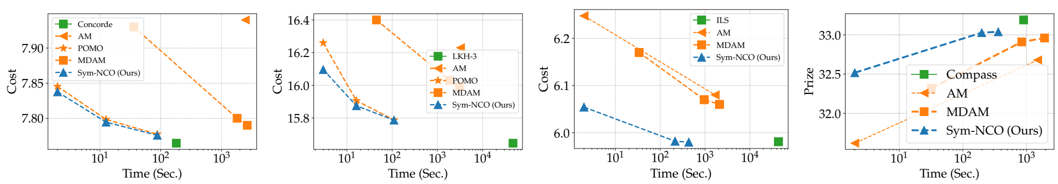
5.4 Experimental results

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



(a) PointerNet (b) AM (c) POMO

Figure 3: 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 4: 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.

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.

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.

Real-world TSP. We evaluate POMO and Sym-NCO on TSPLIB [39]. 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. 3.

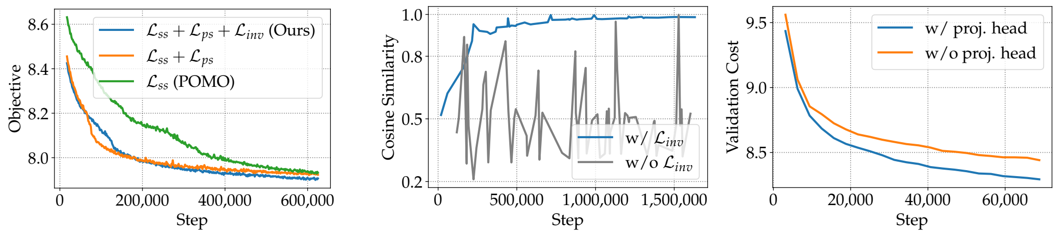
Time-performance analysis for multi-starts Multi-starts is a common method that improves the solution qualities while requiring a longer time budget. We use the rotation augments [21] to produce multiple inputs (i.e., starts). As shown in Fig. 4, Sym-NCO achieves the Pareto frontier for all benchmark datasets. In other words, Sym-NCO exhibits the best solution quality among the baselines within the given time consumption.

6 Discussion

6.1 Soft invariant learning vs. hard constraint invariant learning

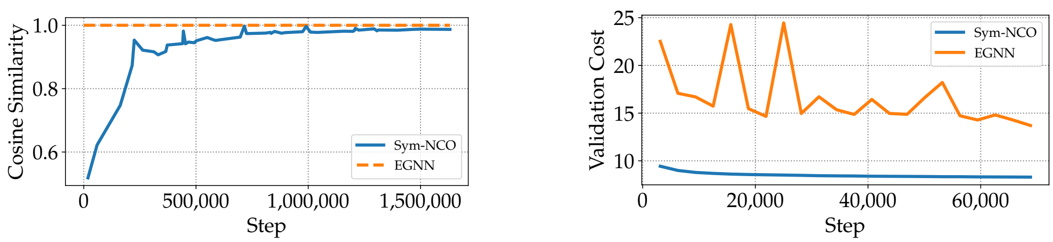
This paper presents a novel invariant learning scheme that imposes the problem symmetricity on NCO via regularization (i.e., *soft invariant learning*). Furthermore, we devise $\mathcal{L}_{\text{inv}}$ to impose the symmetricity on the hidden representations. In this section, we provide the discussion and ablations of these design choices.

Ablation Study of $\mathcal{L}_{\text{inv}}$ As shown in Fig. 5b, $\mathcal{L}_{\text{inv}}$ increases the cosine similarity of the projected representation (i.e., $g(h)$). We can conclude that $\mathcal{L}_{inv}$ contributes to the performance improvements



(a) Cost curves of the loss designs (b) Cosine similarity curves (c) Projection head ablation results

Figure 5: 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 6: Comparisons of Sym-NCO and EGNN

(see Fig. 5a). We further verify that imposing similarity on h degrades the performance as demonstrated in Fig. 5c. This again proves the importance of maintaining the representation capability of the encoder.

Comparison with EGNN One distinct aspect of Sym-NCO is to impose the symmetricies through the *soft invariant learning*. However, imposing the symmetricitiy through *high constraint invariant learning* by modifying the network architecture is also a viable option. To further understand the effects of the symmetricity-imposing mechanisms, we additionally train the *high constraint invaraint learning* model, EGNN [29], as the encoder. As shown in Fig. 6a, the hard approach (i.e., EGNN) strictly enforces the symmetricities. Nevertheless, we observed that EGNN significantly underperforms than Sym-NCO, and fails to converge as shown in Fig. 6b. We suspect that the performance difference originates mainly from the restricted network architecture of EGNN.

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 (this work solves $N < 250$). Hence, we expect that the transfer- [40], curriculum- [41], 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 the non-Euclidean spaces such as asymmetric TSP. We also leave finding the universal symmetricities of non-Euclidean COPs for future research.

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] 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.

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

[10] 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.

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

[12] 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.

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

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

[15] 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.

[16] 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.

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

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

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

[20] MohammadReza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takac. Reinforcement learning for solving the vehicle routing problem. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 9839–9849. Curran Associates, Inc., 2018.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

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

[27] 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.

[28] 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.

[29] 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.

[30] 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.

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

[32] 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.

[33] 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.

[34] 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.

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

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

[37] 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.

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

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

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

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

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]