Symmetric Exploration in Combinatorial Optimization is Free!

Hyeonah Kim[1] **Minsu Kim**[1] **Sungsoo Ahn**[2] **Jinkyoo Park**[1]
[1]Korea Advanced Institute of Science and Technology (KAIST)
[2]Pohang University of Science and Technology (POSTECH)
{hyeonah_kim, min-su, jinkyoo.park}@kaist.ac.kr
{sungsoo.ahn}@postech.ac.kr

Abstract

Recently, deep reinforcement learning (DRL) has shown promise in solving combinatorial optimization (CO) problems. However, they often require a large number of evaluations on the objective function, which can be time-consuming in real-world scenarios. To address this issue, we propose a "free" technique to enhance the performance of any deep reinforcement learning (DRL) solver by exploiting symmetry without requiring additional objective function evaluations. Our key idea is to augment the training of DRL-based combinatorial optimization solvers by reward-preserving transformations. The proposed algorithm is likely to be impactful since it is simple, easy to integrate with existing solvers, and applicable to a wide range of combinatorial optimization tasks. Extensive empirical evaluations on NP-hard routing optimization, scheduling optimization, and de novo molecular optimization confirm that our method effortlessly improves the sample efficiency of state-of-the-art DRL algorithms. Our source code is available at https://github.com/kaist-silab/sym-rd.

1 Introduction

Combinatorial optimization (CO) is an important mathematical field that aims to find the optimal combination in discrete space. Applications of CO include optimization of vehicle routes [1], schedules [2], hardware design[3] and molecular structures [4]. Important CO problems are typically computationally intractable to solve exactly using domain-agnostic solvers like mathematical programming. Thus, researchers have resorted to designing problem-specific heuristics for solving such problems. However, the heuristics are problem-specific, so one has to design a new heuristic with expensive expert knowledge when encountering a new problem. For example, a plethora of heuristics exist for solving the vehicle routing problem, e.g., LKH3 [5], but they are inapplicable to the optimization of molecular structure.

Recently, deep reinforcement learning (DRL) has drawn significant attention as a domain-agnostic strategy to solve CO problems. The main promise of DRL algorithms is that they do not necessitate expert-designed labeled data or problem-specific knowledge to design solvers. They have shown impressive performance even for well-studied problems like vehicle routing (e.g., [6–9]) and job scheduling (e.g., [10–12]). One can expect such DRL solvers to be most powerful when encountering a less-studied CO problem where sophisticated heuristics do not exist.

However, many previous works have not accounted for sample efficiency, assuming that calculating the cost function is cheap. It is important to consider sample efficiency as a crucial metric in real-world applications, where cost functions cannot be evaluated in a closed form, requiring expensive simulations to evaluate solutions, e.g., drug discovery [13], and network design [14]. In such scenarios, the number of evaluations (i.e., reward calls) is restricted, so models need to be trained

ICML 2023 Workshop: Sampling and Optimization in Discrete Space (SODS 2023).

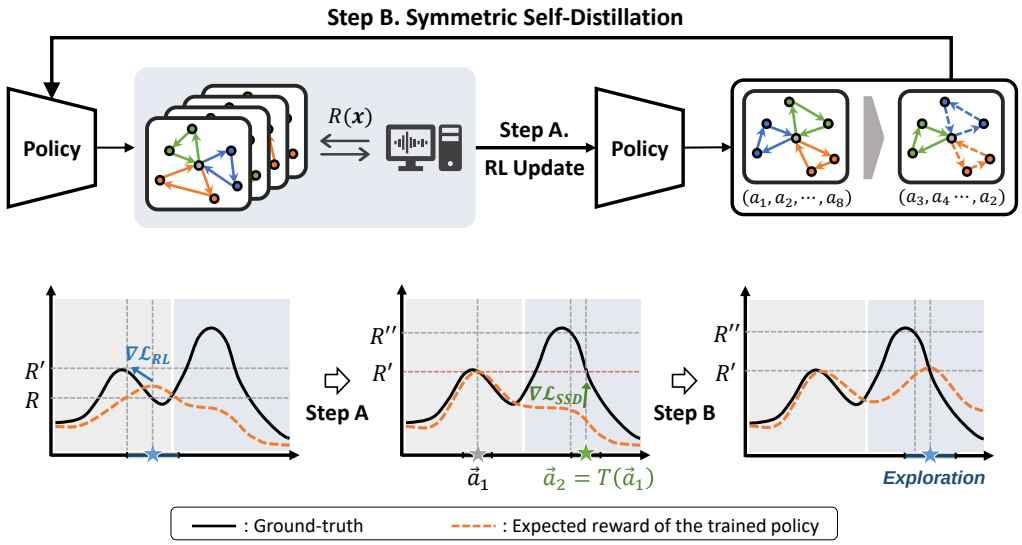


Figure 1: The illustration of SymRD. The DRL policy are encouraged to visit a novel solution state in symmetric spaces using solution-preserving transformations.

sample-efficiently. Recent benchmarks for molecular optimization demonstrate that some advanced DRL methods, e.g., generative flow networks (GFlowNet) [15], often fail to outperform classical methods when sample efficiency is restricted [16].

Contribution. This paper proposes symmetric reinforcement distillation (SymRD) which improves sample efficiency for DRL on a broad range of CO problems. Our key idea is leveraging the symmetric nature of CO by bootstrapping the policy in the symmetric space, which can help the policy to explore a new region without additional reward computation. We demonstrate the broad applicability and performance for various sample-efficient tasks, including Euclidean and non-Euclidean CO problems, and de novo molecular optimization.

At a high level, our method comprises two stages as illustrated in Fig. 1. In the first step, the agent is trained with conventional DRL algorithms, e.g., AM [6] for TSP or REINVENT [17] for molecular optimization. Next, the agent is trained to imitate symmetric solutions generated from itself in the symmetric self-distillation step. We remark that the symmetric self-distillation step does not require additional reward evaluation. The sequential steps in our method serve distinct roles in terms of exploration, as depicted in Fig. 1. The RL update step explores the overall space to discover high-reward trajectories. On the other hand, the symmetric self-distillation explores symmetric trajectories giving the same solution, naturally the same reward, as the explored high-rewarded trajectories in the previous RL update step.

Several DRL approaches have leveraged the symmetric nature of CO. Representatively, the Policy Optimization with Multiple Optima (POMO) [7] and Symmetric Neural CO (Sym-NCO) [8] force to roll out multiple samples in different symmetric spaces. However, in these works, explorations of symmetric spaces are not free, as they require reward evaluation for all symmetric samples. In addition, equivariant DRL methods reduce the search space by employing neural networks that give equivariant representation for symmetric state-action pairs. However, they do not consider the solution symmetries induced by the Markov decision process of CO. Our method is novel in that we allow for the generation of samples through symmetric transformations while incurring no additional cost, as these samples are subject to the same reward. Hence, symmetric exploration in combinatorial optimization is *free*.

We empirically demonstrate how our method significantly improves the sample efficiency of various tasks: (asymmetric) TSP, CVRP, flow-shop scheduling and de novo molecular optimization. The experimental results show that our method is advantageous even when compared to competitive DRL-based CO solvers, including solvers considering symmetric CO properties.

2 Related Works

2.1 Symmetric deep reinforcement learning for combinatorial optimization

Building on the success of the attention model (AM) [6], a Transformer-based model for combinatorial optimization, several works have proposed to improve performances, especially by leveraging the symmetric nature of CO. This is exemplified by the Policy Optimization for Multiple Optima (POMO) [7] and Symmetric Neural Combinatorial Optimization (Sym-NCO) [8]. These methods sample multiple trajectories from a single problem by forcing N heterogeneous starting point based on the TSP's cyclic symmetries [7] or giving symmetric input noise to make wide exploration [8]. Then, they evaluate the REINFORCE baseline using multiple sampled trajectories, which requires reward evaluation for each sample. POMO and Sym-NCO focus on designing an effective REINFORCE baseline using symmetries in CO rather than symmetric explorations. Despite the potential benefits of these methods, they suffer from poor sample efficiency due to the significant computational burden associated with extensive reward evaluation.

Separately, the generative flow network (GFlowNet) [15], which employs a directed acyclic graph (DAG) to represent the combinatorial space in CO problems, was proposed. The DAG structure elegantly models trajectory symmetries, where multiple action trajectories are mapped to identical states (i.e., nodes in DAG), such as molecular segments. GFlowNet utilizes a TD-like balancing loss, where the in-flow and out-flow should be the same for any given DAG node. However, GFlowNet suffers from sample inefficiency due to low-quality exploration in the early stages of training.

2.2 Equivariant deep reinforcement learning

Equivariant DRL has also been extensively studied in recent years [18–22]. This approach reduces search space by cutting out symmetric space using equivariant representation learning, such as employing equivariant neural networks [23–25]. Consequently, it leads to better generalization and sample efficiency. For example, Equivariant representations for RL (EqR) employs equivariant parameterization of state and action and equivariant latent transition models to exploit symmetries [18]; its performances are verified on the sample-efficient Atari. However, in CO, permuted order of actions can lead to the same final state (i.e., solution symmetries). Therefore, it is not straightforward to design DRL models that are equivariant to the order of decisions. Being different from these approaches, we focus on handling symmetries in decision space by exploring the symmetric space without restrictions on network structure. Therefore, employing equivariant DRL methods with our method is one of the possible options when guaranteeing exact equivariance is crucial.

3 Symmetric Reinforcement Distillation (SymRD)

Target problem. Our method aims to improve deep reinforcement learning (DRL) for solving combinatorial optimization where it is expensive to evaluate the objectives. Specifically, we consider combinatorial optimization as maximization over the black-box function $f(\boldsymbol{x})$ over a discrete set $\mathcal{X}$, i.e., $\max_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x})$. To solve this problem, we formulate the construction and evaluation of the solution as a Markov decision process (MDP). In the MDP, we let each state s describe the problem context and an incomplete solution. Then a policy $\pi(a|s)$ decides a transition between states with an action a to update the incomplete solution described by the state s. We assume that the transition is deterministic, i.e., the next state is decided by some transition function t.[1] This also implies that a complete sequence of actions $\vec{a} = (a_1, \dots, a_T)$ starting from an initial state s_1 fully describes an episode of the MDP. The detailed formulations for each task are provided in Appendix D.

We further impose two conditions on the MDP that exploit the prior knowledge about combinatorial optimization problems considered in this work. First, we assume the reward is episodic, i.e., given a terminated action-state trajectory $\tau = (s_1, a_1, \dots, a_{T-1}, s_T)$ associated with a solution $\boldsymbol{x}$, the reward $R(s_T) = f(\boldsymbol{x})$ and $R(s_t) = 0$ for $t < T$. The next condition is about how the action space $\mathcal{A}_t$ for action a_t made at each state s_t only consists of actions that generate a valid solution for the combinatorial optimization.

[1] For example, in the context of TSP, a transition can be represented simply by adding the action (i.e., newly visited cities) to the current state (i.e., a set of previously visited cities), denoted as $s' \leftarrow s \cup \{a\}$.

Overview of SYMRD. Our approach, coined Symmetric Reinforcement Distillation (SYMRD), improves the sample efficiency of DRL for combinatorial optimization by exploiting transformed samples without additional reward computation. The key idea is to exploit the existence of solution-preserving transformation for generating novel action trajectories. Our approach enhances sample efficiency in two ways: (1) additionally training *for free* by imitating self-generated solutions and (2) encouraging exploration in a symmetric region to *escape from low-reward regions*.

Our method repeats the following two steps:

Step A. Train the (factorized) policy $\pi_\theta(\vec{a}|s_1) = \prod_{t=1}^{T} \pi_\theta(a_t|s_t)$ using a conventional episodic reinforcement learning algorithm.

Step B. Generate action trajectory $\vec{a}$ using greedy rollout. Train the policy to imitate the action trajectories sampled from random solution-preserving transformation of $\vec{a}$.

Intuitively, our **Step A** is designed to encourage the policy to exploit the high-rewarded space via reinforcement learning. In contrast, **Step B**, called symmetric self-distillation, aims to promote exploration of the symmetric space without evaluating the episodic reward $R(\vec{a})$.

Note that the DRL model and training method in **Step A** are not restricted, allowing the application of various DRL algorithms for episodic tasks as base DRL methods. For instance, we utilize AM and REINVENT as base DRL methods for TSP and molecular optimization, respectively; we provide details for the base methods in Appendix F. Following the RL update, we sample a transformed action trajectory using the current policy $\pi_\theta(\cdot|s_1)$ and a solution-preserving transformation policy to get more samples for free. In the following sub-sections, we describe the details of a solution-preserving transformation policy and the self-distillation step.

3.1 Solution-preserving transformation policy

Here, we provide an explicit characterization of the solution-preserving transformation policy used for our algorithm. A solution-preserving transformation gives another action trajectory that induces the same solution as a given action trajectory. To begin with, we introduce a non-injective function C that maps an action trajectory $\vec{a} = (a_1, \dots a_T)$ to its corresponding solution $\boldsymbol{x}$ given initial state. Such a mapping allows for defining symmetry between action trajectories with respect to the solution.

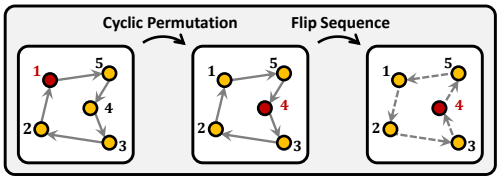
Definition 1 (Symmetric action trajectories)**.** A pair of action trajectories $\vec{a}^1$ and $\vec{a}^2$ given initial state are symmetric if they induce the same solution $\boldsymbol{x} \in \mathcal{X}$, i.e., if $C(\vec{a}^1) = C(\vec{a}^2) = \boldsymbol{x}$.

We also let $\vec{\mathcal{A}}_{\boldsymbol{x}}$ denote a set of symmetric action trajectories that induce the sample solution $\boldsymbol{x}$. Next, we define a *solution-preseving transformation policy* as a probability distribution which generates an action trajectory $\vec{a}$ given the associated solution $\boldsymbol{x}$.

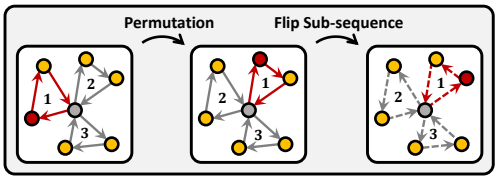
Definition 2 (Solution-preserving transformation policy)**.** We define a solution-preserving transformation policy, denoted by $p_{\text{sym}}(\vec{a}|\boldsymbol{x})$, as the conditional probability of generating an action trajectory $\vec{a}$ given a solution $\boldsymbol{x}$ such that $C(\vec{a}) = \boldsymbol{x}$. Note that the probability distribution $p_{\text{sym}}(\vec{a}|\boldsymbol{x})$ is supported on the action-trajectory space $\vec{\mathcal{A}}_{\boldsymbol{x}}$.

Symmetries in routing tasks. Within the context of TSP, a solution denotes a cycle (i.e., a route) without a designated starting point. Conversely, an action trajectory signifies the sequential order in which cities are visited, with the first and last cities being explicitly defined. Thus, symmetric trajectories are obtained by cyclically shifting k positions to the left or right as illustrated in Fig. 2a. Furthermore, in symmetric TSP, the reversed order of visiting sequence also gives a symmetric action trajectory. On the other hand, CVRP assumes multiple vehicles that start from the depot and return. Therefore, symmetric trajectories are obtained by flipping each sub-route (i.e., each vehicle's route). Note that cyclic permutations do not give symmetric trajectories; instead, permuting vehicle indices by rearranging the order of sub-routes results in symmetric trajectories like Fig. 2b.

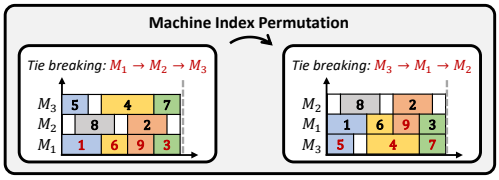
Symmetries in others tasks. In flexible flow-shop scheduling problems (FSSP), idle machines select the next jobs. Symmetries are induced due to the tie-breaking rules, the pre-defined order of action selections. Thus, we define symmetric transformations as the permutation of orders of tie-breaking like Fig. 2c. In de novo molecular optimization, string-based molecular representation (SELFreferencIng Embedded Strings; SELFIES [26]) is employed. We define symmetric action



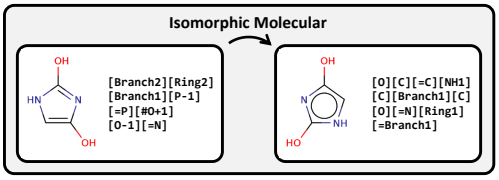
(a) Symmetric transformation in TSP.



(b) Symmetric transformation in CVRP.



(c) Symmetric transformation in FFSP.



(d) Symmetric transformation in MolOpt.

Figure 2: Examples of symmetric transformations in various CO problems.

sequences (i.e., the symmetric SELFIES) as the SELFIES representing isomorphic molecular, which possess the same connectivity of atoms. More details are provided in Appendix E.

3.2 Symmetric self-distillation

The symmetric self-distillation process involves behavior cloning of symmetrically transformed action trajectories $\vec{a}^1, \ldots, \vec{a}^L \sim p_{\text{sym}}(\cdot|\boldsymbol{x}, s_1)$. Here, $\boldsymbol{x}$ is the corresponding solution of an action trajectory sampled from the current training policy, i.e., $\boldsymbol{x} = C(\vec{a})$, where $\vec{a} \sim \pi_\theta(\vec{a}|s_1)$. The symmetric self-distillation loss function is derived as follows:

$$\mathcal{L}_{\text{SSD}} = -\sum_{i=1}^{L} \log \pi_\theta(\vec{a}^i|s_1).$$

Notably, we employ an iterative update procedure, whereby the model is updated to minimize the reinforcement learning loss $\mathcal{L}_{RL}$, and the self-distillation loss $\mathcal{L}_{SSD}$ in turn.

Our symmetric self-distillation significantly benefits from additionally exploring confidence regions on the symmetric space for free. Moreover, since the symmetric action trajectory may have a significant edit distance[2] from the original trajectory, we can explore regions that are likely to have high rewards but are far from the current regions. For instance, the reversed TSP tour (i.e., flipped action sequence) has a vast edit distance from the original but is mapped to the same solution.

Theorem 1. Consider a distribution $\pi_\theta(\vec{a}|s_1)$ over the action trajectory from a state s_1, which describes the problem context. Let $U_{\boldsymbol{x}}(\vec{a}|\boldsymbol{x}, s_1)$ denote an uniform distribution over $\vec{\mathcal{A}}_{\boldsymbol{x}}$. Then the entropy of $\pi(\vec{a}|s_1)$ is upper-bounded as by that of $p_{\text{sym}}(\vec{a}|s_1)$:

$$\begin{aligned}\mathcal{H}(\pi_\theta(\vec{a}|s_1)) &= \mathcal{H}(\pi_\theta(\boldsymbol{x}|s_1)) + \mathbb{E}_{\boldsymbol{x}\sim\pi_\theta(\boldsymbol{x}|s_1)}\mathcal{H}(\pi_\theta(\vec{a}|\boldsymbol{x}, s_1)) \\ &\le \mathcal{H}(\pi_\theta(\boldsymbol{x}|s_1)) + \mathbb{E}_{\boldsymbol{x}\sim\pi_\theta(\boldsymbol{x}|s_1)}\mathcal{H}(U_{\boldsymbol{x}}(\vec{a}|\boldsymbol{x}, s_1)) = \mathcal{H}(U(\vec{a}|s_1)),\end{aligned}$$

where $U(\vec{a}|s_1) = \sum_{\boldsymbol{x}\in\mathcal{X}} U_{\boldsymbol{x}}(\vec{a}|\boldsymbol{x}, s_1)\pi_\theta(\boldsymbol{x}|s_1)$.

Proof. See Appendix B for the entire proof.

According to Theorem 1, the entropy of the training policy π_θ depends on the entropy of the distribution that generates action $\vec{a}$ given input s_1. This distribution represents the backward probability of multiple trajectories originating from a common solution $\boldsymbol{x}$. Maximum entropy exploration is achieved when the distribution is uniform. Therefore, to facilitate maximum entropy exploration in the symmetric self-distillation process, we can choose a uniform distribution for the solution-preserving transformation policy $p_{\text{sym}}(\vec{a}|\boldsymbol{x}, s_1)$ as part of the process: $p_{\text{sym}}(\vec{a}|\boldsymbol{x}, s_1) = U_{\boldsymbol{x}}(\vec{a}|\boldsymbol{x}, s_1)$.

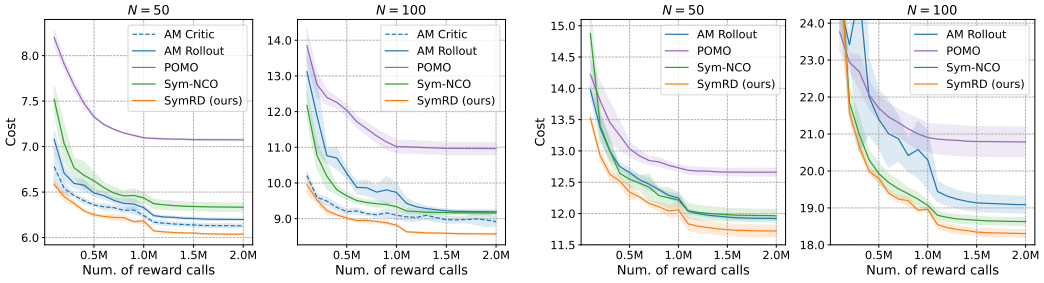
Ideal scenario. Our approach is especially effective when a high-rewarded solution is close to a symmetric action trajectory with a large edit distance. Precisely, $d(\vec{a}^1, \vec{a}^2)$ is sufficiently large and

$$\max_{\vec{a}\in B_\epsilon(\vec{a}^2)} R(C(\vec{a})) \gg \max_{\vec{a}\in B_\epsilon(\vec{a}^1)} R(C(\vec{a})), \quad \text{where } B_\epsilon(\vec{a}') = \{\vec{a}|d(\vec{a}', \vec{a}) \le \epsilon\}.$$

[2] *Edit distance* is a measure of similarity between two sequences of characters or symbols, defined as the minimum number of operations required to transform one sequence into the other, e.g., *Hamming distance*.

Table 1: Experimental results on sample efficient Euclidean CO problems.

	Method	$N = 50$		$N = 100$	
		$K = 200$K	$K = 2$M	$K = 200$K	$K = 2$M
TSP	AM Critic	6.541 ± 0.075	6.129 ± 0.021	9.600 ± 0.090	8.917 ± 0.115
	AM Rollout	6.708 ± 0.077	6.199 ± 0.014	11.891 ± 1.008	9.193 ± 0.053
	POMO	7.910 ± 0.055	7.074 ± 0.010	12.766 ± 0.358	10.964 ± 0.171
	Sym-NCO	7.035 ± 0.209	6.334 ± 0.045	10.776 ± 0.362	9.159 ± 0.056
	SymRD (ours)	$\mathbf{6.450 \pm 0.053}$	$\mathbf{6.038 \pm 0.005}$	$\mathbf{9.521 \pm 0.098}$	$\mathbf{8.573 \pm 0.019}$
CVRP	AM Rollout	13.366 ± 0.199	11.921 ± 0.026	23.414 ± 0.238	19.088 ± 0.232
	POMO	13.799 ± 0.310	12.661 ± 0.065	22.939 ± 0.245	20.785 ± 0.403
	Sym-NCO	13.406 ± 0.204	12.215 ± 0.124	21.860 ± 0.422	18.630 ± 0.106
	SymRD (ours)	$\mathbf{12.922 \pm 0.071}$	$\mathbf{11.721 \pm 0.093}$	$\mathbf{21.582 \pm 0.149}$	$\mathbf{18.304 \pm 0.109}$



(a) TSP (b) CVRP

Figure 3: Validation cost over computation budget on euclidean CO problems.

In such a scenario (also illustrated in Fig. 1), the exploration guidance offered by the symmetric self-distillation provides an opportunity to break out from the low-reward region.

4 Experiments

The experiments in this paper cover various sample-efficient tasks in Euclidean and non-Euclidean combinatorial optimization, as well as de novo molecular optimizations. Note that we assume the expensive black-box reward function in sample-efficient tasks. In Euclidean CO tasks, the features of variables, such as their two-dimensional coordinates, satisfy Euclidean conditions (e.g., cost coefficients are defined as Euclidean distances). On the other hand, non-Euclidean CO problems lack these constraints, necessitating the encoding of higher-dimensional data, such as a distance matrix. We conduct experiments on both with different DRL methods. Lastly, we carry out experiments on the well-known benchmarks in de novo molecular optimization, which aims to find molecules that maximize a certain property (i.e., score function) on combinatorial chemical space. Note that FFSP requires 1 - 2 days, and others require a few hours with a single NVIDIA A100 GPU per experiment.

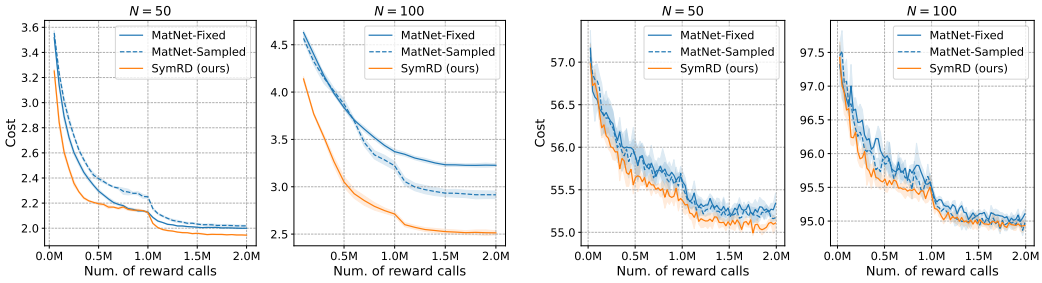
Experimental Setting. In sample-efficient tasks, the reward evaluations are expensive, leading to their computation time taking most of the training time. Therefore, we evaluate performances with limited number of reward evaluation following [16]. For base DRL methods, we employ the best-performing DRL methods and follow reported hyperparameters for the model in their original paper. More details are provided in Appendix A.

4.1 Euclidean CO

Tasks. Following previous DRL literature in CO, we select two representative routing tasks – TSP and CVRP – as Euclidean CO tasks. The TSP aims to find the shortest Hamiltonian cycle that visits every city and returns to the starting city. The CVRP assumes multiple salesmen (i.e., vehicles) with limited carrying capacity; thus, if the capacity is exceeded, the vehicle must return to the depot.

Table 2: Experimental results on sample efficient non-Euclidean CO problems.

	Method	$N = 50$		$N = 100$	
		$K = 200$K	$K = 2$M	$K = 200$K	$K = 2$M
ATSP	MatNet-Fixed	3.139 ± 0.024	2.000 ± 0.002	4.400 ± 0.040	3.227 ± 0.016
	MatNet-Sampled	3.235 ± 0.021	2.019 ± 0.005	4.324 ± 0.036	2.915 ± 0.040
	SymRD (ours)	$\mathbf{2.845 \pm 0.039}$	$\mathbf{1.945 \pm 0.003}$	$\mathbf{3.771 \pm 0.012}$	$\mathbf{2.513 \pm 0.022}$
FSSP	MatNet-Fixed	56.350 ± 0.170	55.341 ± 0.118	96.461 ± 0.206	95.107 ± 0.072
	MatNet-Sampled	56.347 ± 0.234	55.172 ± 0.032	96.256 ± 0.140	94.978 ± 0.055
	SymRD (ours)	$\mathbf{56.104 \pm 0.125}$	$\mathbf{55.110 \pm 0.061}$	$\mathbf{96.030 \pm 0.132}$	$\mathbf{94.934 \pm 0.051}$



(a) ATSP (b) FFSP

Figure 4: Validation cost over computation budget on non-Euclidean CO problems.

Baselines and metric. We compare our method with four baseline methods, including AM with critic baseline, AM with greedy rollout baseline [6], POMO [7], and Sym-NCO [8]. We measure the average cost on the validation dataset over training samples K to evaluate sample efficiency. Note that the validation dataset consists of 10,000 instances randomly generated following [6]. We report the average and standard deviation of results with four independent random training seeds.

Results. The results in Table 1 and Fig. 3 indicate that SymRD consistently outperforms baseline methods in terms of achieving the lowest cost over the training budget. Note that ours employs the AM with critic baseline for TSP and Sym-NCO with the reduced number of augmentations for CVRP. As depicted in Table 1, the most significant improvement over the base DRL models is observed in TSP100, with a percentage decrease of 3.86%, and CVRP50, with a percentage decrease of 4.04%. While POMO and Sym-NCO consider the symmetric nature of CO, the required number of samples cancels out the benefits. In contrast, our method utilizes the symmetric pseudo-labels generated via the training policy for free, enabling the policy to explore the symmetric space without increasing the number of required samples. As a result, SymRD successfully improves sample efficiency.

4.2 Non-Euclidean CO

Tasks. Based on the work of Kwon *et al.* [12], we have selected two benchmark tasks, namely the asymmetric TSP (ATSP) and flexible flow-shop scheduling problems (FSSP). The ATSP is non-Euclidean TSP where the distance matrix could be non-symmetric, i.e., $\text{dist}(i, j) \neq \text{dist}(j, i)$, where i and j indicate cities. The FSSP is an important scheduling problem that assigns jobs to multiple machines to minimize total completion time.

Baselines and metric. As a baseline, we employ Matrix Encoding Network (MatNet) proposed to solve non-Euclidean CO. We compare ours with two versions of MatNet: MatNet-Fixed and MatNet-Sampled. MatNet-Fixed, the original version, explores N heterogeneous starting points of trajectories, while MatNet-Sampled explores less than N number of multiple trajectories with sampling strategy. Similar to the Euclidean CO tasks, we compute the average cost on a validation dataset consisting of 1,000 instances while tracking the cumulative number of training samples. The experiments are conducted with four independent random seeds.

Table 3: Experimental results on sample efficient molecular optimization (AUC-10).

Oracle	REINVENT	MolDQN	GFlowNet	SymRD (ours)
albuterol_similarity	0.871 ± 0.023	0.356 ± 0.014	0.523 ± 0.068	**0.914 ± 0.019**
amlodipine_mpo	0.622 ± 0.015	0.342 ± 0.011	0.458 ± 0.009	**0.634 ± 0.021**
celecoxib_rediscovery	0.661 ± 0.114	0.110 ± 0.008	0.363 ± 0.023	**0.662 ± 0.063**
deco_hop	0.639 ± 0.008	0.558 ± 0.003	0.596 ± 0.003	**0.644 ± 0.005**
drd2	**0.978 ± 0.009**	0.037 ± 0.008	0.807 ± 0.054	**0.978 ± 0.006**
fexofenadine_mpo	0.751 ± 0.006	0.506 ± 0.006	0.720 ± 0.013	**0.770 ± 0.012**
gsk3b	0.843 ± 0.011	0.276 ± 0.012	0.702 ± 0.004	**0.862 ± 0.040**
isomers_c7h8n2o2	0.923 ± 0.025	0.564 ± 0.080	0.606 ± 0.192	**0.938 ± 0.021**
isomers_c9h10n2o2pf2cl	**0.787 ± 0.038**	0.490 ± 0.075	0.269 ± 0.227	0.772 ± 0.079
jnk3	0.644 ± 0.108	0.116 ± 0.008	0.000 ± 0.000	**0.666 ± 0.138**
median1	0.361 ± 0.010	0.142 ± 0.008	0.230 ± 0.006	**0.369 ± 0.008**
median2	0.268 ± 0.011	0.096 ± 0.007	0.189 ± 0.004	**0.270 ± 0.008**
mestranol_similarity	**0.670 ± 0.023**	0.203 ± 0.025	0.374 ± 0.013	0.647 ± 0.022
osimertinib_mpo	0.832 ± 0.007	0.681 ± 0.012	0.798 ± 0.005	**0.835 ± 0.003**
perindopril_mpo	0.542 ± 0.019	0.259 ± 0.024	0.460 ± 0.015	**0.543 ± 0.010**
qed	**0.943 ± 0.000**	0.781 ± 0.021	0.933 ± 0.002	0.942 ± 0.001
ranolazine_mpo	0.778 ± 0.012	0.061 ± 0.021	0.693 ± 0.013	**0.797 ± 0.010**
scaffold_hop	0.539 ± 0.019	0.416 ± 0.012	0.477 ± 0.006	**0.579 ± 0.037**
sitagliptin_mpo	0.423 ± 0.157	0.037 ± 0.029	0.126 ± 0.023	**0.546 ± 0.047**
thiothixene_rediscovery	0.513 ± 0.019	0.110 ± 0.007	0.333 ± 0.022	**0.549 ± 0.043**
troglitazone_rediscovery	**0.364 ± 0.020**	0.138 ± 0.008	0.202 ± 0.006	0.359 ± 0.016
valsartan_smarts	**0.000 ± 0.000**	**0.000 ± 0.000**	**0.000 ± 0.000**	**0.000 ± 0.000**
zaleplon_mpo	**0.535 ± 0.016**	0.092 ± 0.054	0.407 ± 0.018	0.534 ± 0.025
Avg.	0.630	0.277	0.446	**0.644**

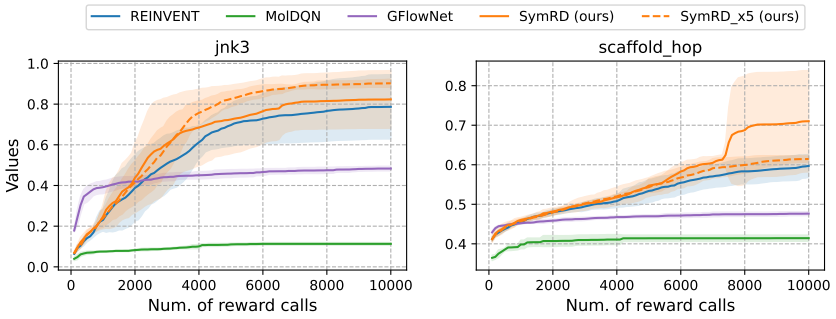


Figure 5: The average TOP-10 values over computation budget.

Results. The superior performance of SymRD over MatNet-Fixed and MatNet-Sampled is demonstrated in both Table 2 and Fig. 4. We employ MatNet-Sampled as a base DRL method for both tasks and use the same number of multi-starting in ours and MatNet-Sampled. Notably, SymRD outperforms MatNet-Sampled by a significant margin in the case of ATSP, with a performance gap of about 12% at $N = 100, K = 200$K, where SymRD achieves 3.771 and MatNet-Sampled achieves 4.324. However, the improvement in the case of FFSP is relatively modest compared to ATSP, which we attribute to the fact that FFSP has fewer symmetries than ATSP, as detailed in Appendix E.

4.3 De novo molecular optimization

Tasks. We employ practical molecular optimization (PMO) [16], which is the official sample efficiency benchmark for de novo molecular optimization. PMO contains 23 tasks based on different score functions called Oracles; a task is a CO problem that maximizes the given score function, such as QED [27], DRD2 [28] and JNK3 [29]. For example, QED measures drug safety, while DRD2 and JNK3 measure bioactivities against their corresponding disease targets. In the PMO benchmark, reward evaluations are limited up to 10,000.

Baselines and metric. Our approach was compared to several RL-based methods, including REINVENT [17], which is considered state-of-the-art (SOTA) in the PMO benchmark, as well as MolDQN

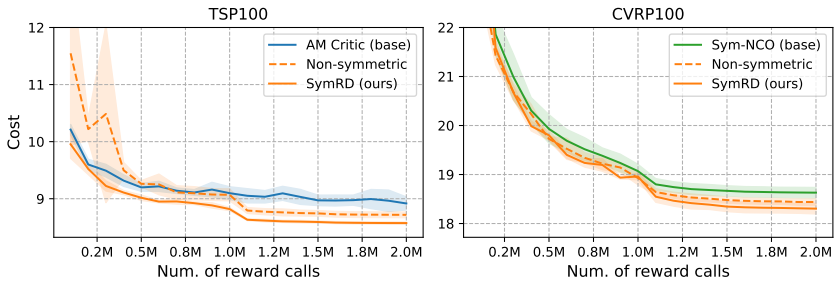


Figure 6: Ablation study for the symmetric self-distillation.

Table 4: Experimental results of measuring trajectory symmetries on the TSP.

		$N = 50$		$N = 100$	
Method	Trained Size	Avg. Cost (↓)	L1 Dist. (↓)	Avg. Cost (↓)	L1 Dist. (↓)
AM Critic	50	6.1153	144.4978	8.8801	315.5329
SymRD (ours)	50	**6.0458**	**30.4985**	**8.726**	**49.8098**
AM Critic	100	6.6262	275.9329	9.1065	569.7021
SymRD (ours)	100	**6.1988**	**32.5435**	**8.6004**	**42.7087**

[30] and GFlowNet [15]. To ensure compliance with chemical constraints such as the octet rule, we employed the SELFIES version of REINVENT. The performance of the methods is evaluated based on the area under curve (AUC) to consider a combination of optimization ability and sample efficiency. The AUC of the top 10 average performance is mainly reported since it is important to find distinct molecular candidates to progress to later stages of development in drug discovery.

Results. As shown in Table 3 and Fig. 5, SymRD outperforms 18 out of 23 tasks, achieving an average AUC-10 score of 0.644, higher than the SOTA method, REINVENT whose average score is 0.630. In Fig. 5, we also report the results of adjusting the distillation period, denoted as 'SymRD_x5', which distills symmetric trajectories every five RL updates to encourage further exploration in **Step A**. It is noteworthy that GFlowNet is also a symmetric exploration method that uses a DAG structure and flow matching to model symmetric action trajectories terminated at an identical state. While the primary purpose of SymRD and GFlowNet is similar, our approach can be straightforwardly embedded into SOTA models (e.g., REINVENT).

4.4 Ablation Study

Components verification. According to the results presented in Fig. 6, each component of the model contributes to the improved sample efficiency observed in our approach. The ablation study showed that self-distillation without symmetries, labeled as 'Non-symmetric', improved sample efficiency but sometimes resulted in unstable performance due to overfitting. However, the inclusion of symmetric transformations of pseudo labels prevented overfitting and resulted in stable performance improvements. We also believe that the symmetric transformation provides a structured inductive bias, enhancing the robustness of self-distillation. Thus, the combination of self-distillation and symmetric transformations can result in stable and efficient performance enhancements.

Measuring symmetries. To validate the ability of our method to enforce trajectory symmetries, we measure the difference between the log-likelihoods of an action sequence produced by the policy and the log-likelihoods of its symmetric trajectories. We utilize the L1 distance metric to quantify this difference, denoted as 'L1 Dist.' in Table 4. The results demonstrate that our approach achieves significantly lower L1 distances compared to the baseline in TSP, indicating its efficacy in enforcing trajectory symmetries. Additionally, our method exhibits robust performance when evaluated on out-of-training distributions.

Maximum entropy exploration. The experimental results show that employing uniform distribution as a solution-preserving transformation policy improves sample efficiency; see Appendix C.

5 Conclusion

This study proposes a new approach called SYMRD to enhance the sample efficiency of deep reinforcement learning (DRL) for combinatorial optimization (CO) problems. Our approach improves sample efficiency by additionally exploring symmetric decision space *for free*. The experimental results demonstrate the superior performance of SYMRD over state-of-the-art methods, such as REINVENT in the de novo molecular design benchmark.

Limitation. Our randomized solution-preserving transformation policy may struggle to cover every possible symmetric space when the space complexity is vast. To overcome this limitation, future research could focus on developing a learnable solution-preserving transformation policy that selectively emphasizes specific symmetric regions. This adaptive policy would suggest symmetric trajectories during training, facilitating more efficient exploration.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2022R1A6A1A03052954).

References

[1] Abdelkader Sbihi and Richard W Eglese. Combinatorial optimization and green logistics. *Annals of Operations Research*, 175(1):159–175, 2010.

[2] Runwei Cheng, Mitsuo Gen, and Yasuhiro Tsujimura. A tutorial survey of job-shop scheduling problems using genetic algorithms—I. representation. *Computers & industrial engineering*, 30(4):983–997, 1996.

[3] Haeyeon Kim, Minsu Kim, Joungho Kim, and Jinkyoo Park. Collaborative distillation meta learning for simulation intensive hardware design. *arXiv preprint arXiv:2205.13225*, 2022.

[4] 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.

[5] Keld Helsgaun. An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University*, 12 2017.

[6] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2018.

[7] 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.

[8] Minsu Kim, Junyoung Park, and Jinkyoo Park. Sym-NCO: Leveraging symmetricity for neural combinatorial optimization. In *Advances in Neural Information Processing Systems*, 2022.

[9] Jiwoo Son, Minsu Kim, Hyeonah Kim, and Jinkyoo Park. Meta-SAGE: Scale meta-learning scheduled adaptation with guided exploration for mitigating scale shift on combinatorial optimization. *arXiv preprint arXiv:2306.02688*, 2023.

[10] Cong Zhang, Wen Song, Zhiguang Cao, Jie Zhang, Puay Siew Tan, and Xu Chi. Learning to dispatch for job shop scheduling via deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1621–1632, 2020.

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

[12] Yeong-Dae Kwon, Jinho Choo, Iljoo Yoon, Minah Park, Duwon Park, and Youngjune Gwon. Matrix encoding networks for neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 34:5138–5149, 2021.

[13] Nathan Brown, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher. GuacaMol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3):1096–1108, 2019.

[14] Longsheng Sun, Mark H Karwan, and Changhyun Kwon. Robust hazmat network design problems considering risk uncertainty. *Transportation Science*, 50(4):1188–1203, 2016.

[15] Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394, 2021.

[16] Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor Coley. Sample efficiency matters: a benchmark for practical molecular optimization. *Advances in Neural Information Processing Systems*, 35:21342–21357, 2022.

[17] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):1–14, 2017.

[18] Arnab Kumar Mondal, Vineet Jain, Kaleem Siddiqi, and Siamak Ravanbakhsh. EqR: Equivariant representations for data-efficient reinforcement learning. In *International Conference on Machine Learning*, pages 15908–15926. PMLR, 2022.

[19] 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.

[20] Arnab Kumar Mondal, Pratheeksha Nair, and Kaleem Siddiqi. Group equivariant deep reinforcement learning. *arXiv preprint arXiv:2007.03437*, 2020.

[21] Dian Wang and Robin Walters. $SO(2)$-equivariant reinforcement learning. In *International Conference on Learning Representations*, 2022.

[22] Andreea Deac, Théophane Weber, and George Papamakarios. Equivariant MuZero. *arXiv preprint arXiv:2302.04798*, 2023.

[23] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

[24] Maurice Weiler and Gabriele Cesa. General $E(2)$-Equivariant steerable CNNs. *Advances in Neural Information Processing Systems*, 32, 2019.

[25] Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. $E(n)$ equivariant graph neural networks. In *International conference on machine learning*, pages 9323–9332. PMLR, 2021.

[26] Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1(4):045024, 2020.

[27] G Richard Bickerton, Gaia V Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90–98, 2012.

[28] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):1–14, 2017.

[29] Yibo Li, Liangren Zhang, and Zhenming Liu. Multi-objective de novo drug design with conditional graph generative model. *Journal of cheminformatics*, 10:1–24, 2018.

[30] Zhenpeng Zhou, Steven Kearnes, Li Li, Richard N Zare, and Patrick Riley. Optimization of molecules via deep reinforcement learning. *Scientific reports*, 9(1):1–10, 2019.

[31] George B Dantzig and John H Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.

[32] Jens Lysgaard, Adam N Letchford, and Richard W Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100(2):423–445, 2004.

[33] Joseph J Rotman. *An introduction to the theory of groups*, volume 148. Springer Science & Business Media, 2012.

[34] David Weininger. SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

A Experimental Setting

In this section, we provide details of the experimental setting for each task. It is noteworthy that we set the base DRL models following the provided code.[3][4][5][6] We adjust learning rate decay at 1M and 1.5M samples. As shown in Table 5, We use the same distillation scaler value for all tasks except for FFSP, which has a different scale of reward value. On the other hand, the number of transformations, L, can be chosen based on the available computation budget; we set $L = 1$ for all experiments.

Table 5: Hyperparameters

	TSP	CVRP	ATSP	FFSP	MolOpt
Common Training Parameters					
Batch size	100	100	100	100	64
Epoch size	10,000	10,000	10,000	10,000	-
Validation size	10,000	10,000	10,000	10,000	-
Learning rate scheduler	MultiStepLR with $\gamma = 0.1$				-
Symmetric Self-Distillation					
Distillation scaler	0.001	0.001	0.001	0.01	0.001
Transformation width (L)	1	1	1	1	1
Base DRL method	AM Critic	Sym-NCO	MatNet	MatNet	REINVENT

B Proof for Theorem 1

Consider $\pi_\theta(\vec{a}|s_1)$ as the distribution over the action trajectory from a state s_1 which describes the problem context. Let $\vec{\mathcal{A}}_{\boldsymbol{x}}$ denote the space of action-trajectories associated with the solution $\boldsymbol{x}$.

$$
\begin{aligned}
\mathcal{H}(\pi_\theta(\vec{a}|s_1)) &= -\sum_{\vec{a}\in\vec{\mathcal{A}}} \pi_\theta(\vec{a}|s_1)\log\pi_\theta(\vec{a}|s_1) \\
&= -\sum_{\boldsymbol{x}\in\mathcal{X}}\sum_{\vec{a}\in\vec{\mathcal{A}}_{\boldsymbol{x}}} \pi_\theta(\vec{a}|s_1)\log\pi_\theta(\vec{a}|s_1) \\
&= -\sum_{\boldsymbol{x}\in\mathcal{X}}\sum_{\vec{a}\in\vec{\mathcal{A}}_{\boldsymbol{x}}} \pi_\theta(\vec{a}|\boldsymbol{x},s_1)\pi_\theta(\boldsymbol{x}|s_1)\left(\log\pi_\theta(\vec{a}|\boldsymbol{x},s_1)+\log\pi_\theta(\boldsymbol{x}|s_1)\right) \\
&= \mathcal{H}(\pi_\theta(\boldsymbol{x}|s_1)) + \mathbb{E}_{\boldsymbol{x}\sim\pi_\theta(\boldsymbol{x}|s_1)}\mathcal{H}(\pi_\theta(\vec{a}|\boldsymbol{x},s_1)) \\
&\le \mathcal{H}(\pi_\theta(\boldsymbol{x}|s_1)) + \mathbb{E}_{\boldsymbol{x}\sim\pi_\theta(\boldsymbol{x}|s_1)}\mathcal{H}(U_{\boldsymbol{x}}(\vec{a}|\boldsymbol{x},s_1)),
\end{aligned}
$$

where $U_{\boldsymbol{x}}(\vec{a}|\boldsymbol{x},s_1)$ is a uniform distribution over action-trajectories associated with the solution $\boldsymbol{x}$. The third equality stems from the fact that $\pi_\theta(\vec{a}|s_1) = \pi_\theta(\vec{a},\boldsymbol{x}|s_1)$ since $\boldsymbol{x}$ is fixed given $\vec{a}$. One can show that the final upper-bound is the entropy of distribution obtained from replacing $\pi_\theta(\boldsymbol{x}|s_1)$ by $U_{\boldsymbol{x}}(\vec{a}|\boldsymbol{x},s_1)$, i.e., distribution from sampling $p_{sym}(\vec{a}|s_1)$.

[3] AM: https://github.com/wouterkool/attention-learn-to-route
[4] Sym-NCO: https://github.com/alstn12088/Sym-NCO
[5] MatNet: https://github.com/yd-kwon/MatNet
[6] REINVENT: https://github.com/wenhao-gao/mol_opt

C Maximum Entropy Exploration on Symmetric Space

We empirically verify Theorem 1 by employing different solution-preserving transformations. We compare SYMRD with identical transformation and uniform transformation distribution. To make results clear, we use $L = 10$, i.e., each trajectory is transformed to 10 other trajectories.

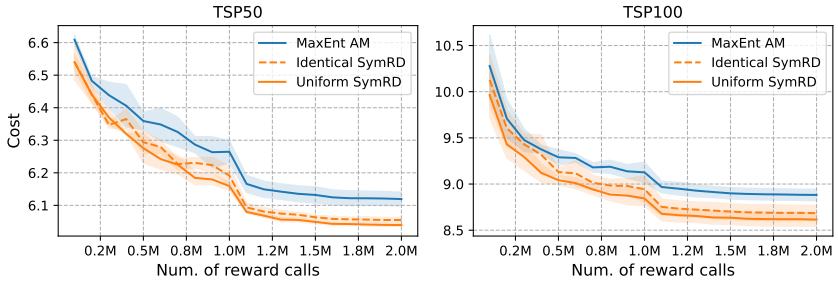


Figure 7: Validation cost improvement over the computation budget

As a baseline, we implement AM with entropy bonus (denoted in 'MaxEnt AM' in the figures). In MaxEnt AM, we add entropy bonus to the reward to use REINFORCE loss as follows:

$$\nabla \mathcal{L} = \mathbb{E}_{\pi_\theta} \left[\left(f(\boldsymbol{x}) - b(s_1) - \alpha \sum_{i=1}^{N} \pi_\theta(\vec{a}|s_1) \log \pi_\theta(\vec{a}|s_1) \right) \nabla \log \pi_\theta(\vec{a}|s_1) \right],$$

where $b(s_1)$ is the critic baseline [6], and we set $\alpha = 0.01$. The results show that both identical SYMRD and uniform SYMRD outperform the baseline, AM with entropy bonus (denoted in 'MaxEnt AM' in the figures). Additionally, maximum entropy exploration (i.e., employing uniform distribution for the solution-preserving transformation policy) achieve better performance than identical transformation policy.

D Task Formulations

For each task, we provide brief explanation, integer programming (IP) formulation, and Markov decision process (MDP) formulation.

D.1 Travelling salesman problems

In travelling salesman problems (TSP), a salesman must visit each of N cities exactly once and then return to the starting city. Given the travelling distance from city i to j, $d_{ij} \geq 0$ for each i and j, TSP finds a minimum length of route. In symmetric TSP, the distance are symmetric, i.e., $d_{ij} = d_{ji}$, while asymmetric TSP is a relaxed version of this assumption.

IP formulation. We let the decision variable x_{ij} represent whether the salesman visit j right after vising i. Given a directed complete graph $G = (V, E)$ where V is the set of cites and E is the set of fully connected edges between cities, a TSP is formulated as follows:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in E} d_{ij} x_{ij} & \\
\text{subject to} \quad & \sum_{j\in V} x_{ij} = 1 & \forall i \in V \\
& \sum_{i\in V} x_{ij} = 1 & \forall j \in V \\
& \sum_{(i,j)\in \delta^+(S)} x_{ij} \geq 1 & \forall S \subset V, S \neq \emptyset \\
& x_{ij} \in \{0, 1\} & \forall i, j \in V
\end{aligned}
$$

where $\delta^+(S)$ is a set of outgoing edges from S to $V \setminus S$. Additionally, we let $x_{ij} = x_{ji}$ for the symmetric TSP.

Markov decision process.

- State s_t is composed with locations of N cities $\{v_i\}_{i=1}^N$ and previously selected city indices $(a_1, ..., a_{t-1})$: $s_t = \{\{v_i\}_{i=1}^N; a_1, ..., a_{t-1}\}$
- Action a_t is selection of unvisited city index: $a_t \in \{1, ..., N\} \setminus \{a_1, \ldots, a_{t-1}\}$
- Transition $s_{t+1} = t(s_t, a_t)$ is simply set union of them: $s_{t+1} = t(s_t, a_t) = s_t \cup \{a_t\}$
- Reward $R(s_N)$ is defined with negative tour-length of terminal state of s_N: $R(s_N) = -\left(d(a_N, a_1) + \sum_{t=1}^{N-1} d(a_{t+1}, a_t)\right)$

D.2 Capacitated vehicle routing problems

The capacitated vehicle routing problem (CVRP) [31] aims to find the routes with the minimum cost for K vehicles where: (i) each customer in V_C must be served by one vehicle; (ii) vehicle serve a set of customers whose total demand does not exceed capacity Q; and (iii) every route starts and ends at the depot, usually index as 0.

IP formulation. We let the edge variable x_{ij} represent the number of travels between vertices i and j following [32]. Since the edges are undirected, we let $x_{ij} = x_{ji}$. Given a subset of customers $S \subseteq V_C$, we let $\delta(S)$ denote a set of crossing edges which connect S and $V_C \setminus S$. The *two-index formulation* of CVRP is:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in E} d_{ij}x_{ij} & \\
\text{subject to} \quad & x(\delta(\{i\})) = 2 & \forall i \in V_C \\
& x(\delta(S)) \geq 2b(S) & \forall S \subseteq V_C \\
& x_{ij} \in \{0,1\} & \forall 1 \leq i < j \leq |V_C| + 1 \\
& x_{0j} \in \{0,1,2\} & \forall j \in V_C,
\end{aligned}
$$

where $b(S)$ is the minimum number of required vehicles to serve the customers in S.

Markov decision process.

- State s_t is composed with locations of N cities $\{v_i\}_{i=1}^N$ and one depot v_0 and previously selected city indices $(a_1, \ldots, a_{t-1})$: $s_t = \{\{v_i\}_{i=1}^N; a_1, \ldots, a_{t-1}\}$
- Action a_t is selection of unvisited city index (depot $a = 0$ can be visited multiple times): $a_t \in \{0\} \cup \{1, \ldots, N\} \setminus \{a_1, \ldots, a_{t-1}\}$. Note the start and terminal action is restricted to select depot city: $a_1 = a_T = 0$.
- Transition $s_{t+1} = t(s_t, a_t)$ is simply set union of them: $s_{t+1} = t(s_t, a_t) = s_t \cup \{a_t\}$
- Reward $R(s_T)$ is defined with negative tour-length of terminal state of s_T: $R(s_T) = -\sum_{t=1}^{T-1} d(a_{t+1}, a_t) = -\left(\sum_{t=1}^{T-1} ||v_{a_{t+1}} - v_{a_t}||_2\right)$

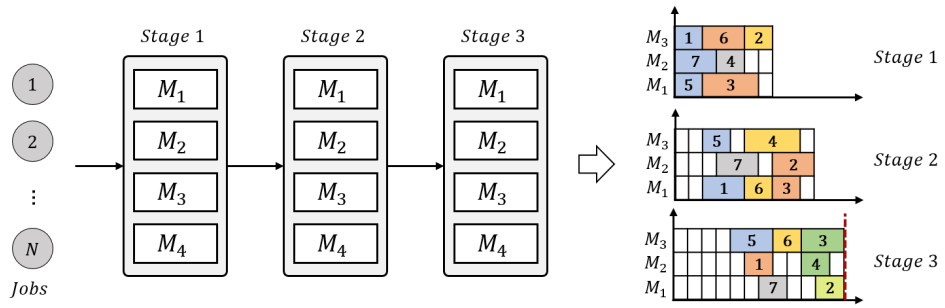


Figure 8: An example of flexible flow shop problems

D.3 Flexible flow shop problems

The flexible flow shop problems (FFSP), also called a hybrid flow shop problems, is a generalization of two scheduling problems, the flow shop scheduling problem and the parallel machines scheduling problem. In FFSP, there are jobs with varying processing times, which have to be processed following pre-defined order of operations (i.e., stage). There are multiple machines for each stage, so they can process jobs in parallel. The FFSP aims to find scheduling, i.e., assigning jobs into machines with a specific order, that minimizes completion time.

IP formulation. We follow the formulation in [12]. Decision variable x_{ijm} is 1 if job j is assigned to machine m in stage i, and y_{ilj} is 1 if job l is processed earlier than job j in stage i. Let C_{ij} denote the completion time of job j in state i and p_{ijm} denote the processing time of job j with machine m in stage i. Lastly, N, S, and M_i denote the number of jobs, the number of stages, and the number of machines in stage i, respectively.

$$
\begin{aligned}
\text{minimize} \quad & \max_{j=1,\dots,N} C_{Sj} \\
\text{subject to} \quad & \sum_{m=1}^{M_i} x_{ijm} = 1 && i = 1,\dots,S, \quad j = 1,\dots,N \\
& y_{ijl} = 0 && i = 1,\dots,S, \quad j = 1,\dots,N \\
& \sum_{j=1}^{N}\sum_{l=1}^{N} y_{ijl} = \sum_{m=1}^{M_i} \max\left(\sum_{j=1}^{N} x_{ijm} - 1, 0\right) && \forall i = 1,\dots,S \\
& y_{ijl} \le \max\left(\max_{m}(x_{ijm} + x_{ilm}) - 1, 0\right) && i = 1,\dots,S, \quad j,l = 1,\dots,N \\
& \sum_{l=1}^{N} y_{ijl} \le 1 && i = 1,\dots,S, \quad j = 1,\dots,N \\
& \sum_{l=1}^{N} y_{ilj} \le 1 && i = 1,\dots,S, \quad j = 1,\dots,N \\
& C_{1j} \ge \sum_{m=1}^{M_i} p_{1jm} x_{1jm} && j = 1,\dots,N \\
& C_{ij} \ge C_{i-1j} + \sum_{m=1}^{M_i} p_{ijm} x_{ijm} && i = 2,\dots,S, \quad j = 1,\dots,N \\
& C_{ij} + M(1 - y_{ilj}) \ge C_{il} + \sum_{m=1}^{M_i} p_{ijm} x_{ijm}, && i = 1,\dots,S, \quad j = 1,\dots,N
\end{aligned}
$$

where M is sufficiently large value.

Markov decision process. The completion time of job j in state i is denoted as C_{ij}, and the processing time of job j with machine m in stage i is represented as p_{ijm}. Additionally, N, S, and M_i represent the total number of jobs, stages, and machines in stage i, respectively.

- State s_t is composed with processing time matrix $\boldsymbol{D}^i$ for each stage i and previously assigned job indices for each machine.
- Action a_t is assigning a job to the idle machine or choosing skip: $a_t \in \{1, \ldots, N\} \setminus \{a_1, \ldots, a_{t-1}\} \cup \{\varnothing\}$
- Transition $s_{t+1} = t(s_t, a_t)$ is simply set union of them: $s_{t+1} = t(s_t, a_t) = s_t \cup \{a_t\}$
- Reward $R(s_T)$ is defined with the maximum completion time (i.e., makespan): $R(s_T) = -\max_j C_{Sj}$

D.4 De novo molecular optimization

De novo molecular optimization involves the search for a molecule $\boldsymbol{x}$ that maximizes a pre-defined black-box scoring function $f(\boldsymbol{x})$. This scoring function can represent various properties, such as binding activity with a specific target protein or a measure of toxicity, among others. Unlike classical combinatorial optimization tasks like the Traveling Salesman Problem (TSP), the formulation of de novo molecular optimization cannot rely on mathematical programming due to the nature of $f(\boldsymbol{x})$ being a black-box function without a closed form. Furthermore, evaluating $f(\boldsymbol{x})$ is typically computationally expensive in practice. To overcome this limitation and ensure accessibility, researchers in the field often employ simplified (approximated) score functions that allow for computational tractability.

To construct a DRL framework for de novo molecular optimization, we employ the MDP (Markov Decision Process) formulation of REINVENT-SELFIES [16]. This approach utilizes SELFIES [26] strings as a means to generate valid representations of molecular graphs.

Markov decision process.

- State s_t is sequence of SEFILES tokens a: $s_t = \{a_1, ..., a_{t-1}\}$
- Action a_t is a selection of SEFLIES tokens. Note the start and terminal action is restricted to select the GO token and EOS token correspondingly.
- Transition $s_{t+1} = t(s_t, a_t)$ is simply set union of them: $s_{t+1} = t(s_t, a_t) = s_t \cup \{a_t\}$.
- Reward $R(s_T)$ is defined with score function of $f(\boldsymbol{x})$ where $\boldsymbol{x} = C(\vec{a})$.

E Symmetries of Each Task

We denote an unordered set of elements $e_1, e_2, \ldots, e_n$ as $\{e_1, e_2, \ldots, e_n\}$, then we can define permutations and cycles as below.

Definition 3 (Permutations [33])**.** If Ω is a nonempty set, a *permutation* of Ω is a bijection $\alpha : \Omega \to \Omega$. We denote the set of all permutations of Ω by P_Ω.

Given a rearrangement, a list $e_1, e_2, \ldots, e_n$ with no repetitions of all the elements of $\Omega = \{1, 2, \ldots, n\}$, defines a function $\alpha : \Omega \to \Omega$ by $\alpha(\omega) = e_\omega$ for all $\omega \in \Omega$. Any bijection α can be denoted as follows:

$$\alpha = \begin{pmatrix} 1 & 2 & \cdots & n \\ \alpha(1) & \alpha(2) & \cdots & \alpha(n) \end{pmatrix},$$

where the bottom row is a rearrangement of Ω.

Definition 4 (Cycles [33])**.** Let $e_1, e_2, \ldots, e_n$ be distinct integers between 1 and n. If $\beta \in P_{\{1,2,\ldots,n\}}$ fixes (i.e., $\beta(\omega) = \omega$) the remaining $n - r$ integers and if

$$\beta(e_1) = e_2, \beta(e_2) = e_3, \ldots, \beta(e_{r-1}) = e_r, \beta(e_r) = e_1,$$

then β is a *r-cycle*.

Definition 5 (Flip)**.** Let $e_1, e_2, \ldots, e_n$ be distinct integers between 1 and n. If $\gamma \in P_{\{1,2,\ldots,n\}}$ and $\gamma(e_i) = e_{n+1-i}$, then γ is a flip.

E.1 Symmetries in routing problems

Symmetries in (asymmetric) TSP. Given $\vec{a} = (a_1, \ldots, a_N)$, any N-cycles β_N and its k products (i.e., $\beta_N^k = \beta_N \circ \cdots \circ \beta_N$) give the same route. Especially in symmetric TSP, permuting $\vec{a}$ to the reversed order, i.e., flip, is also gives the symmetric trajectory. Note that products of each symmetric transformation are also symmetric transformations.

Symmetries in CVRP. Let κ_j denote the route of vehicle $j = 1, \ldots K$. Any permutation of routes, i.e., any $\alpha \in P_\mathcal{K}$, where $\mathcal{K} = \{\kappa_1, \ldots, \kappa_K\}$, is mapped to the same solution. Unlike to the TSP route, each route must start at the depot and return to it, so cycles of each route do not give symmetric trajectories, but flip of each route does. Note that products of permutations of routes and flip within a route give symmetric trajectories, as well.

E.2 Symmetries in other tasks

Symmetries in FFSP. In decoding process, the time index t is introduced, and every t, idle machines (i.e, machines without assigned job at t) select a job or 'skip'. The job selection order among idle machines can be arbitrary decided; we denote this order as a *tie-breaking rule* in the main. From the perspective of machines, their schedules are independent on others' schedule when a specific schedule is given. In other words, the whole machine-jobs schedule can be decomposed into the schedules of each machine. Consequently, the different tie-breaking rules does not change each schedule. Precisely, any permutations of machine index, i.e, any $\alpha \in P_\mathcal{M}$, where $\mathcal{M} = \{1, \ldots, M\}$, gives symmetric trajectories. For example, in Fig. 2c, the original order of machines is $1 \to 3 \to 2 \to 1 \to 3 \to 1 \to 2 \to 1 \to 3$, but the order of machines in the symmetric trajectory is $3 \to 1 \to 2 \to 3 \to 1 \to 1 \to 2 \to 3 \to 1$.

Symmetries in molecular optimization. In de novo molecular optimization, we employ a string-based representation, SELFreferencIng Embedded Strings (SELFIES) [26]. SELFIES starts with a specific atom and searches molecular with depth first search to convert molecular into a string. Thus, the different starting point can give different SELFIES even for the same molecular. Furthermore, there can be isomorphic molecular which possess the same connectivity of atoms. We let two SELFIES representations are symmetric if the corresponding molecular is the same or isomorphic. Practically, we get symmetric SELFIES by converting the given SELFIES into molecular and re-converting them to SELFIES, including information about stereochemistry.

F Details of Base DRL Methods

This section provides an overview of the base DRL method for each task. See each paper for more technical details.

F.1 Attention Model for TSP

The attention model (AM) [6] is a widely used DRL method for solving routing problems such as the Traveling Salesman Problem (TSP). The AM employs a transformer-like model that generates encoder embedding based on the city locations and utilizes auto-regressive decoding to determine the city index. Training of the AM involves the REINFORCE method along with rollout baselines. For more technical details, refer to the respective paper.

F.2 Sym-NCO for CVRP

Sym-NCO [8] shares similarities with the AM but focuses on addressing the Capacitated Vehicle Routing Problem (CVRP) using more sophisticated REINFORCE baseline estimators. It takes advantage of the symmetrical nature of CVRP by applying augmentation techniques such as rotation to the input cities, preserving the solution space. It then calculates the average reward from the policy's solutions to multiple augmented problems, which becomes the shared baseline for all samples. This strategy improves the accuracy of the REINFORCE baseline estimation. For further details, refer to the associated paper.

F.3 MatNet for FFSP and ATSP

MatNet [12] extends the AM by incorporating edge-wise considerations and is specifically designed for solving graph-based combinatorial optimization problems like the Flexible Flow Shop Problem (FFSP) and the Asymmetric Traveling Salesman Problem (ATSP). Since these problems involve edges, MatNet employs an edge-wise attention mechanism. The training mechanism follows a policy optimization approach for multiple optima, similar to Sym-NCO's shared baseline strategy. For more detailed information, consult the paper related to MatNet.

F.4 REINVENT for MolOpt

REINVENT [17] is a notable DRL method for de novo molecular optimization, known for its state-of-the-art performance and sample efficiency in benchmark settings [16]. Originally, REINVENT utilized recurrent neural networks (RNN) as policy models for generating SMILES (Simplified Molecular Input Line Entry System) sequences [34]. The training of the RNN policy employed the REINFORCE method, without the support of a prior network similar to the baseline strategy or actor-critic. It is worth noting that [16] proposes variants of REINVENT specifically for generating SEFILES strings, ensuring the validity of the generated molecules.