Symmetric Replay Training: Enhancing Sample Efficiency in Deep Reinforcement Learning for Combinatorial Optimization

Hyeonah Kim [1] **Minsu Kim** [1] **Sungsoo Ahn** [2] **Jinkyoo Park** [1 3]

Abstract

Deep reinforcement learning (DRL) has significantly advanced the field of combinatorial optimization (CO). However, its practicality is hindered by the necessity for a large number of reward evaluations, especially in scenarios involving computationally intensive function assessments. To enhance the sample efficiency, we propose a simple but effective method, called *symmetric replay training (SRT)*, which can be easily integrated into various DRL methods. Our method leverages high-reward samples to encourage exploration of the under-explored symmetric regions without additional online interactions – *free*. Through replay training, the policy is trained to maximize the likelihood of the symmetric trajectories of discovered high-rewarded samples. Experimental results demonstrate the consistent improvement of our method in sample efficiency across diverse DRL methods applied to real-world tasks, such as molecular optimization and hardware design.

1. Introduction

Combinatorial optimization (CO) problems arise across diverse industrial domains, but they are notoriously challenging to solve. In CO (e.g., traveling salesman problems; TSP), the massive discrete solution space often leads to NP-hardness. Furthermore, CO problems in practical scenarios often involve computationally expensive objective functions to evaluate (e.g., a black-box function), introducing significant restrictions on the problem-solving process. Even though solving CO problems with expensive objective functions is frequently found in various fields like drug discovery or hardware design, it has numerous challenges.

Recent advances in deep reinforcement learning (DRL) have drawn significant attention as an alternative way to solving CO problems. Various studies have demonstrated remarkable achievements across broad domains, including routing (Kool et al., 2018; Kwon et al., 2020; Son et al., 2023), scheduling (Park et al., 2023; Kwon et al., 2021), and material design (Olivecrona et al., 2017; Ahn et al., 2020a; Bengio et al., 2023). Especially in constructive methods, where a solution is sequentially built from an empty set, it is beneficial to guarantee feasibility to practical constraints using a masking strategy. Note that in the constructive method, a policy generates a trajectory, and the reward is evaluated in a terminal state, i.e., episodic rewards.

DRL methods generally leverage extensive samples to enhance training stability, assuming reward evaluation is affordable. However, in practice, reward computation can be costly due to several factors: (1) objective functions often cannot be defined analytically, requiring computationally intensive simulations for evaluation, such as in black-box optimization; or (2) objective functions may be analytically defined but still demand additional computation, such as solving another optimization problem (e.g., bi-level optimization). For instance, in molecular optimization, evaluating designed molecules in real-world scenarios involves costly assessments, such as wet lab experiments (Gao et al., 2022). In this regard, reducing the number of reward evaluations is significantly beneficial. This discrepancy raises a straightforward research question: *How can we enhance the sample efficiency of DRL methods for a broad spectrum of CO problems and methods?*

This paper proposes a simple yet effective add-on method, *symmetric replay training (SRT)*, which is a generic method that leverages the solution-symmetric nature of combinatorial space. In DRL for CO, multiple action sequences (i.e., action trajectories) can be mapped to a single combinatorial solution, and the reward is defined on the terminal state (i.e., solution). Previous studies have tried to consider the symmetries in CO, but they utilize numerous samples (Kwon et al., 2020; Kim et al., 2022b) or assume the specific method (Yao

[1]Department of Industrial & Systems Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea [2]Pohang University of Science and Technology (POSTECH), Pohang, South Korea [3]OMELET, Daejeon, South Korea. Correspondence to: Hyeonah Kim <hyeonah_kim@kaist.ac.kr>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

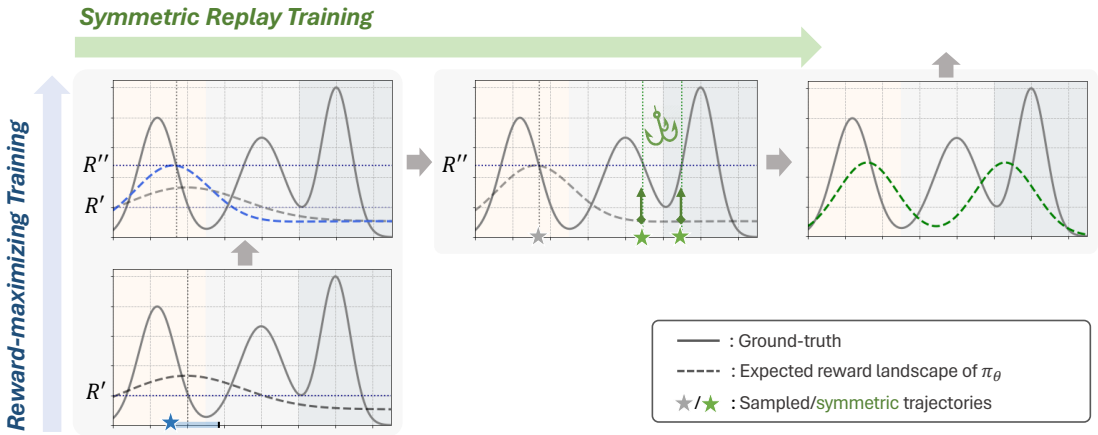

Figure 1. Illustration for two-step training strategies: reward-maximizing training and symmetric replay training.

et al., 2022) or problem (Duan et al., 2022). To effectively utilize symmetries in CO, we suggest a decomposed training process in two steps: reward-maximizing training and symmetric replay training. The two-step training also allows efficient exploration in the vast action trajectory space.

Reward-maximizing training is conducted with a conventional DRL algorithm, which involves the exploration of the high-rewarded action trajectories over the entire action space. Subsequently, symmetric replay training bootstraps the policy to explore symmetric regions without additional reward evaluation, leading to improved sample efficiency. In detail, the high-rewarded action trajectories obtained in the previous step are symmetrically transformed, and then the policy is trained to imitate these symmetric action trajectories. As illustrated in Figure 1, the symmetric replay training promotes the exploration of under-explored regions containing trajectories equivariant to the high-rewarded trajectories collected from reward-maximizing training.

The two decomposed training steps, reward-maximizing and symmetric replay training, are effectively inter-operated: one seeks the high-reward samples, while the other recycles the explored samples to explore symmetric regions *for free*. Symmetric replay training strategically leverages previously high-rewarded trajectories without necessitating additional reward evaluation. Our policy is trained to maximize the likelihood of high-rewarded trajectories in the symmetric region, encouraging exploration of under-explored areas. Additionally, replaying symmetric trajectories incorporates symmetric priors, relieving the overfitting of replay training. Since the policy perceives these symmetric trajectories as heterogeneous, SRT is advantageous in scaling up the replay loop to further utilize collected samples before re-interacting with the environment.

For a better comprehension of symmetric replay training, this paper explores three straightforward strategies: (1) maximum entropy, (2) adversarial, and (3) importance sampling transformations. They provide theoretical and empirical insights for SRT. Note that the maximum entropy transformation is primarily adopted since it is a generic approach with low computational complexity while effectively improving sample efficiency in various tasks.

We empirically demonstrate that symmetric replay training consistently improves the sample efficiency by plugging it into various DRL methods with simple implementation. The experimental results show that plugging our method into the state-of-the-art DRL method can achieve superior performance in hardware design optimization and sample-efficient molecular optimization by adding it to the competitive DRL method. Here is a highlight of our contribution:

1. We propose a generic method, symmetric replay training (SRT), which enhances the sample efficiency of various DRL methods for CO. The proposed method is empirically demonstrated as simple yet effective, by consistently improving various DRL methods, both on-policy and off-policy, in various tasks.

2. We investigate various symmetric transformation strategies for replay training, offering both theoretical and empirical perspectives that could benefit future efforts in SRT-like replay training approaches.

3. We conduct comprehensive experiments with a broad range of DRL methods and various CO tasks. We benchmark the DRL methods from on-policy to off-policy methods across various tasks from classical TSP to practical hardware design optimization, which can contribute to DRL for CO communities.

2. Related Works

2.1. Symmetries in DRL for CO

Several works have been studied to utilize the symmetric nature of combinatorial optimization. Representatively, the

Policy Optimization for Multiple Optima (POMO; Kwon et al., 2020) and Symmetric Neural Combinatorial Optimization (Sym-NCO; Kim et al., 2022b) are proposed. These methods sample multiple trajectories from a single problem by forcing N heterogeneous starting point based on the TSP's cyclic symmetries (Kwon et al., 2020) or giving symmetric input noise to make wide exploration (Kim et al., 2022b). Importantly, they use symmetries to effectively evaluate the REINFORCE baseline, which stabilizes training by effectively reducing the variance. However, they associate extensive reward evaluation, while ours does not require additional sample evaluations to utilize symmetries. Furthermore, the results in Appendix D.7 demonstrate that SRT can further improve the sample efficiency of these methods. The improvements come from the difference that they take account of symmetries in input space, while ours utilize symmetries in output (action trajectories) space.

Separately, Generative Flow Networks (GFlowNets; Bengio et al., 2023), which represents the combinatorial space in CO problems with a directed acyclic graph, were proposed. We discuss the relationship with GFlowNet in Section 4.4.

2.2. Data Augmentation

The symmetric transformation shares a similar concept with data augmentation techniques, such as cropping (Laskin et al., 2020; Yarats et al., 2021), noise injection (Laskin et al., 2020; Fan et al., 2021), and Mixup (Fan et al., 2021; Zhang et al., 2018), all of which can contribute to improved sample efficiency. However, most data augmentation methods focus on continuous space. Notably, recent efforts have been made to adapt data augmentation NLP tasks (Shen et al., 2020; Ren et al., 2021; Feng et al., 2021; Kim et al., 2021b). One of the key ideas is to obtain informative samples while keeping the semantics of original sentences, (Wang et al., 2018; Garg & Ramakrishnan, 2020). Notably, Kim et al. (2021b) introduces a learnable augmentation policy that constructs difficult but not too different samples (DND). As our symmetric transformation preserves the solutions by utilizing the nature of CO tasks, augmenting trajectories does not lose semantics. Note that the adversarial transformation is designed to find difficult samples akin to DND.

In CO, Duan et al. (2022) presented a label-preserving augmentation in contrastive learning for the Boolean satisfiability problem. Additionally, Yao et al. (2022) utilized augmentation, including flipping, in supervised learning for the TSP. Lastly, Kim et al. (2023) suggested an additional loss term using symmetrically augmented data in hardware design. These works suggested data augmentation based on symmetries, which is integrated into a specific learning framework. On the other hand, our paper introduces a generic method to enhance diverse DRL methods across various domains. We empirically show that the separated MLE training of SRT is also beneficial compared to data augmentation training; see Appendix D.3. Note that SRT uses imitation loss, which allows training with augmented trajectories without introducing importance weights, even in on-policy DRL.

2.3. Replay Ratio Scaling

Increasing the number of replay loops is highly related to scaling up the replay ratio, which means the number of parameter updates per environment interaction (Wang et al., 2017; Fedus et al., 2020; D'Oro et al., 2022). Replay ratio is also known as update-to-data (UTD) ratio (Chen & Tian, 2019; Smith et al., 2022). Although the benefits of replay ratio scaling are limited, it has demonstrated improved performance, particularly on well-tuned algorithms (Kielak, 2019; Chen & Tian, 2019; Smith et al., 2022). Recently, D'Oro et al. (2022) achieved better replay ratio scaling with parameter reset strategy by mitigating the loss of ability to generalize on model-free RL. Most replay ratio scaling approaches assume the utilization of replay training, thus off-policy training, and propose how to further increase replay ratio by preventing overfitting or primacy biases (D'Oro et al., 2022; Nikishin et al., 2022). On the other hand, we propose a new replay method both for off-policy and on-policy training in CO. Though SRT demonstrates that it can increase replay ratio less suffering from overfitting, other replay ratio scaling approaches like periodic resetting policy parameters or sophisticated design for experience buffer can be incorporated to enhance sample efficiency further.

We provide further related works in Appendix E.

3. Background

Our method aims to improve the sample efficiency of deep reinforcement learning for solving combinatorial optimization with expensive reward function. Specifically, we consider combinatorial optimization as maximization of the black-box function $f(x)$ over a discrete set $\mathcal{X}$, i.e.,

$$\max_{x \in \mathcal{X}} f(x).$$

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 a subsequence of the action trajectory with problem context $\boldsymbol{c}$, i.e., $s_t = \{(a_1, \ldots, a_{t-1}), \boldsymbol{c}\}$. The initial state corresponds to an empty, i.e., $s_1 = \{\emptyset, \boldsymbol{c}\}$, and the final state corresponds to a complete sequence of actions, i.e., $s_T = \{\vec{a}, \boldsymbol{c}\}$, giving a solution x. Then, a policy $\pi(s'|s)$ decides a transition between states, $s \to s'$, by selecting an action a to be updated to the incomplete solution described by the state s. We assume that the transition is deterministic, meaning that the next state is determined by a specific transition function.

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 associated with a solution x, the reward $R(s_T) = f(x)$ and $R(s_t) = 0$ for $t < T$. Note that the s_T contains the action sequence $\vec{a}$, giving a solution x. The next condition is that the action space $\mathcal{A}_t$ at each state s_t only consists of actions that generate a valid solution.

4. Methodology

Our method, *symmetric replay training (SRT)*, is an add-on method that improves the sample efficiency of DRL for CO by replaying the symmetrically transformed solution. The key idea is to utilize the symmetric natures of CO to generate different trajectories, which induce the same solutions, i.e., solution-preserving transformation; no additional reward evaluations are required as the solution remains unchanged. SRT enhances sample efficiency in two ways: (1) by recycling once-evaluated samples via replay training and (2) by stimulating the exploration of under-explored regions via symmetric trajectories.

Our method repeats the following two policy update loops:

Step A: Reward-maximizing training. Train the (factorized) policy using a conventional episodic reinforcement learning algorithm.

Step B: Symmetric replay training

1. Collect high-rewarded trajectories from Step A.
2. Randomly sample trajectories using symmetric transformation policy.
3. Train the policy by imitating the symmetric trajectories.

Intuitively, our **Step A** is designed to encourage the policy to exploit high-reward samples via reinforcement learning. Then, **Step B** aims to promote exploration of the symmetric space by imitating the collected high-reward samples with symmetric transformation. Note that the strategies for collecting high-reward samples are flexibly employed according to tasks or DRL methods, such as greedy rollout, reward-prioritized sampling, or top-K selection.

4.1. Reward-maximizing Training

Reward-maximizing training corresponds to the original training of deep reinforcement learning (DRL) methods. For simplicity, this step is illustrated using the example of REINFORCE, a widely used DRL algorithm for CO (e.g., Bello et al., 2017; Kool et al., 2018; Kim et al., 2022b). In reward-maximizing training, the policy π_θ is trained for a

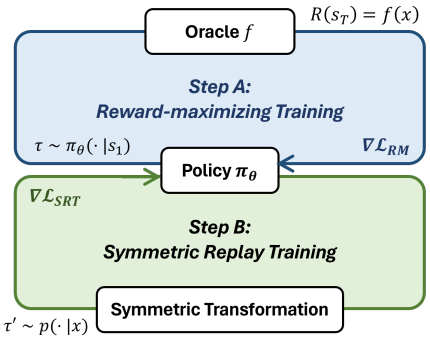

Figure 2. Overview of iterative training

given trajectory $\tau = (s_1, \ldots, s_T)$ using the loss function defined as follows:

$$\mathcal{L}_{\text{RM}} = (R(x) - b) \log \pi_\theta(\tau),$$
$$\text{where } \pi_\theta(\tau) = \prod_{t=1}^{T-1} \pi_\theta(s_{t+1}|s_t).$$

Here, b is a baseline that does not depend on x. It is worth emphasizing that the model and training method in reward-maximizing training are not restricted. Thus, our method allows the application of various DRL methods as a base.

4.2. Symmetric Replay Training

4.2.1. Symmetric transformation policy

The symmetric trajectory policy $p(\tau_{\to x}|x)$ is a probabilistic distribution that samples symmetric trajectories $\tau_{\to x}$, which have an identical terminal state (i.e., an identical solution x). For example, in the traveling salesman problem (TSP), a representative CO problem, shifting the starting points gives the same solution. In the case of TSP with four cities, the sequences 1-2-3-4-1 and 3-4-1-2-3 represent the identical Hamiltonian cycle x.

For a better comprehensive understanding of the symmetric transformation policy, we investigate possible choices of $p(\tau_{\to x}|x)$: maximum entropy, adversarial, and importance sampling transformations.

Maximum entropy transformation policy. By setting $p(\tau_{\to x}|x)$ as a uniform distribution, we can achieve the maximum entropy for the training policy $\pi_\theta(\cdot)$; see Theorem 4.1 in Section 4.3 for theoretical analysis. Notably, the uniform distribution stands out for its simplicity and $\mathcal{O}(1)$ sampling cost (e.g., in TSP, a symmetric trajectory is determined by simply choosing the starting point and direction), easily applicable to various tasks. Therefore, we adopt the maximum entropy policy as a default for our experiments.

Adversarial transformation policy. An alternative option is to sample a trajectory that has the minimum log-

likelihood under $\pi_\theta(\tau)$. Formally, symmetric trajectories are selected as follows:

$$\min_{\tau \in \mathcal{T}_{\to x}} \log \pi(\tau),$$

where $\mathcal{T}_{\to x}$ denotes a set of all trajectories that give a solution x. In the case of TSP, where the number of symmetric trajectories is $2N$, where N is the problem size, the sampling cost is $\mathcal{O}(N)$.

Importance sampling transformation policy. The last option is to sample trajectories based on importance weights, which are set inversely proportional to the log-likelihood. Formally, symmetric trajectories are sampled from

$$p(\tau_{\to x}|x) \propto e^{-\beta \log \pi_\theta(\tau_{\to x})}.$$

Here, β is an inverse temperature, a tunable hyperparameter. Intuitively, it becomes the maximum entropy transformation as $\beta \to 0$, and the adversarial transformation as $\beta \to \infty$; refer to Appendix A.2 for formal proof. Therefore, having $0 < \beta < \infty$ can be considered a compromised approach between the maximum entropy and adversarial transformations. Note that evaluating the importance weight requires $\mathcal{O}(N)$ computations in TSP.

4.2.2. Symmetric replay training

The symmetric replay training involves maximum likelihood estimation of $\tau_{\to x}$ sampled from $p(\tau_{\to x}|x)$. The symmetric replay training loss function is derived as follows:

$$\mathcal{L}_{\text{SRT}}(x) = -\mathbb{E}_{p(\tau_{\to x}|x)}[\log \pi_\theta(\tau_{\to x})] \tag{1}$$

The SRT loss function is formulated to maximize the log-likelihood of the symmetric trajectories. To adjust the scale of the loss function, we introduce a scaling coefficient. Roughly, α is set to make $\mathcal{L}_{\text{SRT}}$ 10 or 100 times smaller than $\mathcal{L}_{\text{RM}}$. Note that the high-reward samples are obtained by greedy rollout or selecting Top-K samples in the mini-batch, and so on.

Our symmetric replay training significantly benefits from additionally exploring the high-rewarded regions on the symmetric space for free. Moreover, we can explore regions that are likely to have higher rewards but are far from the current regions, since the symmetric action trajectory may have a significant edit distance from the original trajectory.[1]

4.3. Interplay of Reward-maximizing and Symmetric Replay Training

This subsection presents an analysis of the interplay between two iterative steps: reward-maximizing training and symmetric replay training. First, we begin with introducing a theorem about the maximization of policy entropy.

Theorem 4.1. *Consider a distribution $\pi_\theta(\tau)$ over the trajectory, and $\pi_\theta(x)$ over its corresponding solutions. Let $U(\tau_{\to x}|x)$ denote an uniform distribution. Then the entropy of $\pi_\theta(\tau)$ can be decomposed and upper bounded as follows:*

$$\begin{aligned}\mathcal{H}(\pi_\theta(\tau)) \\ &= \underbrace{\mathcal{H}(\pi_\theta(x))}_{\textit{Step A}} + \underbrace{\mathbb{E}_{x\sim\pi_\theta(x)}\mathcal{H}(p(\tau_{\to x}|x))}_{\textit{Step B}} \\ &\leq \mathcal{H}(\pi_\theta(x) + \mathbb{E}_{x\sim\pi_\theta(x)}\mathcal{H}(U(\tau_{\to x}|x)).\end{aligned} \tag{2}$$

Proof. See Appendix A.1 for the entire proof.

In Theorem 4.1, the policy entropy $\mathcal{H}(\pi_\theta(\tau))$ is decomposed into two distinct components: the entropy associated with the solution exploration policy, denoted as $\mathcal{H}(p(x))$, and the entropy for the symmetric transformation policy, expressed as $\mathbb{E}_{x\sim p(x)}\mathcal{H}(p(\tau_{\to x}|x))$. This decomposition offers an intuitive illustration of our two-step learning process. In particular, maximizing the second entropy term encourages the search of various symmetric spaces while preserving high-reward solutions. We achieve this by employing uniform symmetric transformation, represented as $p(\tau_{\to x}|x) = U(\tau_{\to x}|x)$. This approach enables us to maximize entropy exploration within the symmetric space, facilitating a more comprehensive search of potential solutions.

4.4. Relationship with GFlowNets

Our symmetric transformation policy $p(\tau_{\to x}|x)$ is similar to GFlowNet's backward policy $P_B(s_t|s_{t+1})$ where both assume the multiple possible (partial) trajectories that leads to identical states: i.e., directed acyclic graph (DAG)-like MDP. While GFlowNets trains policy using a balanced objective with P_B, such as trajectory balance (Malkin et al., 2022), which is known to give slow learning signals when trajectory length is large (Falet et al., 2024), our method makes direct credit assignment into a trajectory drawn from $p(\tau_{\to x}|x)$, which gives faster learning signals. Accordingly, our method significantly improves GFlowNets training efficiency by using our periodic replay training of maximum likelihood estimation in experiments on three tasks.

5. Experiments

This section presents experiments with three distinct settings: a synthetics scenario and two real-world scenarios of hardware design and sample-efficient molecular optimization. For the synthetic setting, we employ traveling salesman problems (TSP), the widely studied CO problems, for precise analyses of the proposed method. We also provide the experiments on other CO problems in Appendix D.7.

In the subsequent sections, we validate the effectiveness of

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

Table 1. Sample efficiency on the synthetic TSP ($N = 50$) with four independent seeds. The average costs (↓) and the standard deviations are reported.

Method	$K = 100$K	$K = 2$M
A2C (Bello et al., 2017)	6.780 ± 0.208	6.129 ± 0.021
A2C + SRT (ours)	$\mathbf{6.586 \pm 0.043}$	$\mathbf{6.038 \pm 0.005}$
PG-Rollout (Kool et al., 2018)	7.138 ± 0.196	6.226 ± 0.026
PG-Rollout + SRT (ours)	$\mathbf{6.879 \pm 0.110}$	$\mathbf{6.131 \pm 0.019}$
PPO (Schulman et al., 2017)	6.771 ± 0.120	6.319 ± 0.110
PPO + SRT (ours)	$\mathbf{6.712 \pm 0.024}$	$\mathbf{6.249 \pm 0.045}$
GFlowNet (Bengio et al., 2023)	6.797 ± 0.078	6.195 ± 0.010
GFlowNet + SRT (ours)	$\mathbf{6.596 \pm 0.030}$	$\mathbf{6.167 \pm 0.013}$

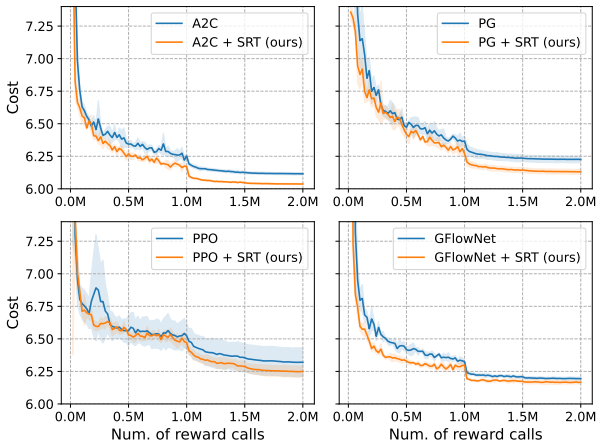

Figure 3. Optimization curves on TSP ($N = 50$)

our method in real-world applications. Firstly, we conduct experiments focused on hardware design optimization, particularly addressing the Decap Placement Problem (DPP), a widely recognized problem within the hardware design domain (Koo et al., 2017; Kim et al., 2021a; Berto et al., 2023). Furthermore, we extend our experiments to the Practical Molecular Optimization (PMO) benchmark (Gao et al., 2022), a well-established benchmark for sample-efficient molecular optimization. All data and codes are available at https://github.com/kaist-silab/symmetric_replay.

5.1. Synthetic Setting: Traveling Salesman Problems

In general, CO problems have closed forms of objective functions, which means the reward evaluations are not expensive. However, we conduct experiments on the TSP synthetic dataset assuming that the number of computing objective function values is limited; this allows more controlled experiments and more precise analysis.

Tasks. Traveling salesman problems (TSP) aim to minimize the distance of a tour that visits all customers and returns to the starting point. In TSP, the distance between

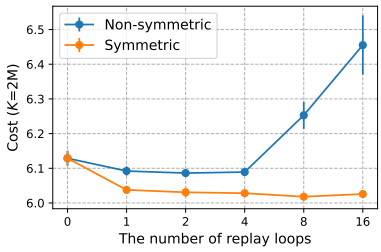

Figure 4. Increasing replay loop

consequent customers is defined as an Euclidean distance. The auto-regressive policy starts from an empty tour and constructs the (partial) tour by iteratively selecting the next visit. In TSP, a solution denotes a cycle (i.e., a route) without a designated starting point. Thus, symmetric trajectories are obtained by cyclically shifting k positions to the left or right. Furthermore, in TSP with Euclidean distance, the reversed order of visiting sequence also gives a symmetric action trajectory. We set the maximum reward calls as 2M.

Experimental setting. Our experiments are conducted with AM architecture from Kool et al. (2018). We employ various RL methods, including policy gradient with actor-critic (Bello et al., 2017), policy gradient with greedy rollout (Kool et al., 2018), Proximal Policy Optimization (Schulman et al., 2017), and a Generative Flow Network (Bengio et al., 2023), and additionally implement SRT on top of the DRL methods to enhance sample efficiency. Note that GFlowNet is an off-policy method. We basically follow the hyperparameter configuration used in AM. For additional parameters of PPO and GFlowNet, we systematically evaluate several combinations to identify the most optimal configuration. The details about implementations are provided in Appendix B. We measure the average costs on the validation dataset over the number of reward calls (K) in training with four independent random seeds. In symmetric replay training, we gather confidence trajectories from the up-to-date policy via greedy rollout, assuming that the greedy solution gives a relatively high reward trajectory.

Does SRT improve sample efficiency? As illustrated in Figure 3 and detailed in Table 1, our approach consistently demonstrates enhanced sample efficiency across various DRL methods. Notably, A2C outperforms the other methods under conditions where the number of available training samples is limited, thereby resulting in the best performance when combined with SRT. The most substantial improvement facilitated by SRT is observed in the case of GFlowNet, where a cost reduction of 3.76% is achieved when the sample size $K = 100$K. In PG-Rollout, a 1.53% cost reduction is realized with SRT when $K = 2$M.

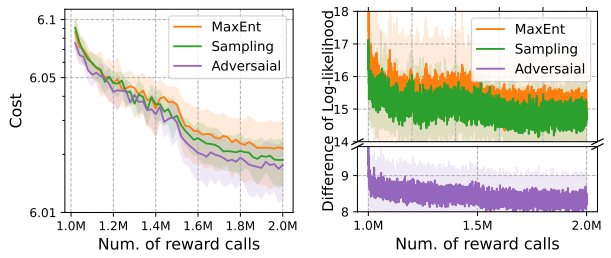

Figure 5. The optimization curve (*left*) and difference between symmetric trajectories (*right*) with different transformation polices.

Overfitting in replay training. In this section, we investigate the extent to which symmetric replay training can increase the number of replay loops without encountering issues of overfitting. To assess the effectiveness of symmetric transformation in replay training, we perform the same experiments without symmetric transformation (the results are denoted as 'Non-symmetric' in Figure 4). As illustrated in Figure 4, our symmetric replay training successfully enhances sample efficiency up to replay loops 16. In contrast, non-symmetric replay training experiences diminished performance when reaching when the loop exceeds 8. This suggests that the symmetric transformation provides trajectories that induce the same solution but are distinctive from the policy's perspective, contributing to mitigating overfitting from replaying the restricted set of repetitive trajectories.

Comparing different strategies of $p(\tau_{\to x}|x)$. We conduct experiments to compare different symmetric transformation policies. The difference of loglikelihood between symmetric trajectories is measured as follows:

$$\frac{1}{|\mathcal{T}_{\to x}|}\sum_{\tau\in\mathcal{T}_{\to x}}\log\pi_\theta(\tau)-\min_{\tau\in\mathcal{T}_{\to x}}\log\pi_\theta(\tau)$$

As illustrated in Figure 5, the adversarial transformation effectively minimizes log-likelihood differences among symmetric trajectories, thereby enhancing sample efficiency. Despite its effectiveness, the adversarial transformation might not be unavailable when the analytical knowledge of symmetric trajectories is not accessible. On the other hand, the maximum entropy transformation is a more generic approach, so it is advantageous for practical scenarios.

Additional experiments. We provide the experimental results to verify the effectiveness of each choice by ablating the components. Moreover, we compare SRT with experience replay. Ours shows superior performance by not introducing importance weight, leading to low variance and allowing replay training of on-policy methods like A2C. All results are provided in Appendix D.

Table 2. Experimental results of DPP on two different PDN environments. The average rewards (↑) and standard deviations are reported with four independent runs.

Method	Chip-package PDN	HBM PDN
A2C (Kool et al., 2018)	9.772 ± 0.823	25.945 ± 0.177
A2C + SRT (ours)	**12.757 ± 0.267**	**26.449 ± 0.094**
PG-Rollout (Kool et al., 2018)	10.240 ± 0.955	25.714 ± 0.122
PG-Rollout + SRT (ours)	**12.601 ± 0.467**	**26.355 ± 0.013**
PPO (Schulman et al., 2017)	9.821 ± 0.411	25.907 ± 0.068
PPO + SRT (ours)	**11.279 ± 0.511**	**26.322 ± 0.141**
GFlowNet (Bengio et al., 2023)	9.740 ± 0.245	25.845 ± 0.103
GFlowNet + SRT (ours)	**12.784 ± 0.186**	**26.469 ± 0.057**

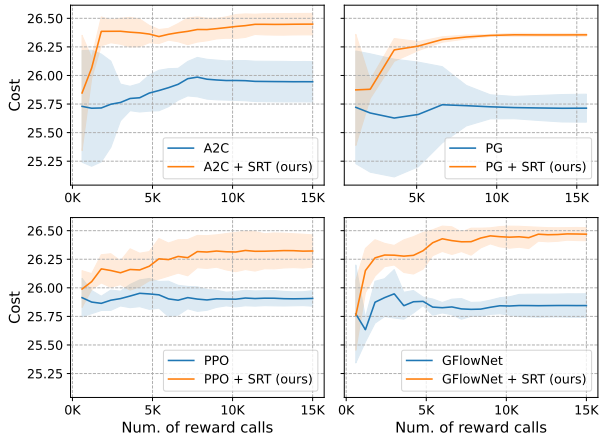

Figure 6. Optimization curves on HBM PDN

5.2. Hardware Design Optimization

Tasks. We employ decoupling capacitor placement problems (DPP), which constitute a fundamental optimization challenge in hardware design. In the context of hardware devices like CPUs and GPUs, a decoupling capacitor (decap) is a critical component responsible for reducing power noises along the power distribution network (PDN). The primary objective of DPP is to identify the optimal arrangement for placing these decaps to maximize the power integrity (PI) objective, which involves computationally expensive evaluations. We tackle two distinct DPP tasks: the chip-package PDN (Koo et al., 2017) and the High Bandwidth Memory (HBM) PDN (Jun et al., 2017) with 15K limited reward calls. These two tasks exhibit variations in their PI landscapes, presenting unique challenges. In DPP, permuting the decision orders of decap yields symmetric trajectories.

Experimental setting. We employ a Device Transformer (DevFormer; Kim et al., 2023) as our base neural architecture, originally designed for offline DPP tasks. In an online optimization context involving interactions with the DPP environment, we integrate well-established DRL methods

(A2C, PG, PPO, and GFlowNet) on DevFormer to maximize effectiveness. Similar to the experiment on TSP, we adjust hyperparameters by examining several combinations since DPP tasks have different reward scales. The details are provided in Appendix B. Lastly, we use higher-than-average samples in the online batch as high-rewarded samples. To verify the effectiveness of our methods, SRT is built on these DRL methods to enhance sample efficiency and compare the resulting rewards.

Results. Addressing DPP tasks proved challenging due to the limited online reward calls available. For instance, the PG-Rollout method exhibited high variances, making it less effective at exploring the DPP solution space. This challenge stems from the inherent symmetry of DPP, where multiple trajectories could lead to identical solutions. Conversely, GFlowNet models, incorporating structured bias to handle solution symmetries, showed improved performance compared to non-symmetric DRL methods like PPO. As demonstrated in Table 2 and Figure 6, significant improvements are observed by applying SRT on top of these methods. The most considerable reward improvement is observed in A2C for chip-package PDN, at 30.54%. In the case of GFlowNet with SRT, exploring focused on the symmetric variants within the high-reward region enhances the sample efficiency by mitigating its underfitting. Notably, the wall-clock comparison results in Appendix D.4 show that SRT effectively enhances performance by achieving significantly lower costs while maintaining similar or slightly increased runtime.

5.3. Practical Molecular Optimization Benchmark

Tasks. We employ practical molecular optimization (PMO; Gao et al., 2022), whose reward evaluations are limited up to 10K, and the goal is achieving the highest score within the limited reward calls. 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` (Bickerton et al., 2012), `DRD2` (Olivecrona et al., 2017) and `JNK3` (Li et al., 2018). For example, `QED` measures drug safety, while the others measure bioactivities against their corresponding disease targets. In de novo molecular optimization, molecules are represented as graphs or strings,[2] which have multiple ways for a single molecule.

Experimental setting. We build SRT on the GFlowNet (Bengio et al., 2023) and REINVENT (Olivecrona et al., 2017). REINVENT is the most competitive DRL method, and GFlowNet is beneficial to generate diverse candidates in the PMO benchmark. Note that REINVENT contains the online experience buffer, so we also collect the replay samples from the experience buffer with the same replay size. For GFlowNet, we set the replay size as the same as the sample batch size. The performance of the methods is evaluated based on the area under the 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 essential to find distinct molecular candidates to progress to later stages of development in drug discovery. Lastly, as a high-rewarded sample to replay, we gather Top-K samples from the entire pool of generated samples up to that point in training.

[2]In this study, we mainly employ SELFreferencIng Embedded Strings (SELFIES; Krenn et al., 2020) for the string-based representation to ensure compliance with chemical constraints.

Table 3. Experimental results on sample efficient molecular optimization. AUC top-10 (↑) is reported with five independent runs. We use oracle ID instead of a name for better readability.

Oracle	GFlowNet	GFlowNet + SRT (ours)	REINVENT	REINVENT + SRT (ours)
#1	0.459 ± 0.028	**0.526 ± 0.022**	0.847 ± 0.021	**0.889 ± 0.008**
#2	0.437 ± 0.007	**0.448 ± 0.010**	0.605 ± 0.017	0.616 ± 0.018
#3	0.326 ± 0.008	**0.345 ± 0.011**	**0.603 ± 0.084**	0.590 ± 0.040
#4	**0.587 ± 0.002**	0.582 ± 0.004	0.630 ± 0.013	**0.650 ± 0.045**
#5	0.601 ± 0.055	**0.796 ± 0.054**	0.953 ± 0.006	**0.960 ± 0.005**
#6	**0.700 ± 0.005**	0.688 ± 0.006	0.735 ± 0.004	**0.751 ± 0.009**
#7	**0.666 ± 0.006**	0.657 ± 0.010	0.800 ± 0.016	**0.828 ± 0.034**
#8	0.468 ± 0.211	**0.928 ± 0.006**	**0.940 ± 0.014**	0.924 ± 0.036
#9	0.199 ± 0.199	**0.628 ± 0.024**	0.842 ± 0.018	**0.858 ± 0.016**
#10	0.442 ± 0.017	**0.505 ± 0.040**	0.574 ± 0.101	**0.665 ± 0.113**
#11	0.207 ± 0.003	**0.211 ± 0.002**	0.350 ± 0.012	**0.351 ± 0.015**
#12	0.181 ± 0.002	0.181 ± 0.002	0.255 ± 0.014	**0.260 ± 0.010**
#13	0.332 ± 0.012	**0.339 ± 0.004**	0.618 ± 0.020	**0.637 ± 0.065**
#14	**0.785 ± 0.003**	0.784 ± 0.003	**0.821 ± 0.007**	0.820 ± 0.005
#15	**0.434 ± 0.006**	0.429 ± 0.010	**0.538 ± 0.030**	0.536 ± 0.015
#16	0.917 ± 0.002	**0.922 ± 0.002**	0.940 ± 0.001	**0.941 ± 0.000**
#17	**0.660 ± 0.004**	0.652 ± 0.008	0.749 ± 0.015	**0.779 ± 0.019**
#18	0.464 ± 0.003	**0.466 ± 0.003**	0.529 ± 0.020	**0.538 ± 0.010**
#19	0.217 ± 0.022	**0.282 ± 0.013**	0.511 ± 0.029	**0.535 ± 0.046**
#20	**0.292 ± 0.009**	0.291 ± 0.007	0.459 ± 0.011	**0.536 ± 0.035**
#21	**0.190 ± 0.002**	0.189 ± 0.004	0.342 ± 0.016	**0.372 ± 0.032**
#22	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
#23	0.353 ± 0.024	**0.398 ± 0.010**	0.512 ± 0.017	**0.514 ± 0.015**
Avg.	0.431	**0.489**	0.615	**0.632**

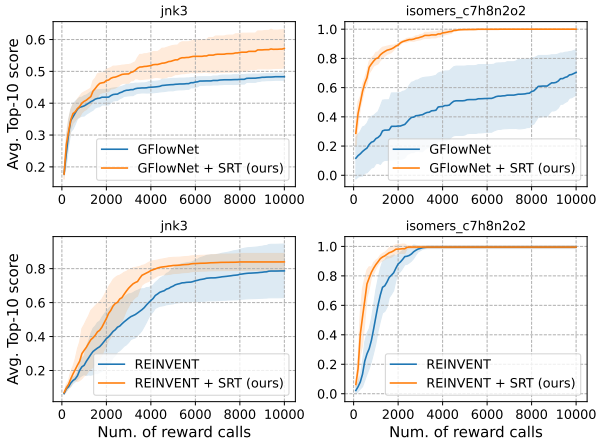

Figure 7. Average score of Top-10 in `jnk3` (Oracle #10) and `isomers_c7h8n2o2` (Oracle #8).

Results. As shown in Table 3, SRT significantly improves GFlowNet and REINVENT by achieving the enhanced AUC in 12 and 18 oracles out of 23, respectively, not only on average; the statistical analyses are provided in Appendix D.5. Note that REINVENT is regarded as a state-of-the-art method in PMO, and SRT also improves 16 oracles out of 23 in the experiment of REINVENT with SMILES strings; see Appendix D.6. In addition, Figure 7 further demonstrates our method's effectiveness by illustrating the average scores of the top 10 molecules in the `JNK3` and `isomer_c7h8n2o2` tasks, revealing a substantial enhancement in sample efficiency.

6. Conclusion

This study proposes a new approach, called *symmetric replay training (SRT)*, to enhance the sample efficiency of DRL methods for practical combinatorial optimization problems. Our approach improves the sample efficiency by reusing the high-rewarded samples from the policy in the symmetric space, which helps explore new regions without additional reward computation. Replay training through symmetric transformations enhances the sample efficiency by effectively increasing the replay ratio while mitigating the adverse effects of overfitting. The proposed method is a generic add-on approach that can synergize with previously proposed constructive DRL methods across various CO problems. Since our method is conducted separately (in Step B), we do not introduce additional restrictions on training or alter the architecture of base DRL methods (in Step A). Another contribution of our method is the utilization of maximizing log-likelihood loss functions and separated training steps. Therefore, SRT does not introduce importance weight, which often leads to large variances, even when applied with on-policy methods. The extensive experiments verify that our method consistently enhances sample efficiency in various DRL methods in real-world benchmarks, like hardware design and molecular optimization.

Impact Statement

This research intersects reinforcement learning and combinatorial optimization, areas that inherently involve various industrial and social impacts. However, we believe that this work does not present any unique societal consequences that necessitate specific emphasis beyond those typically associated with these fields.

Acknowledgments

The authors are grateful to Haeyeon Kim for the help in implementing the hardware design simulator. This work was supported by the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (2022-0-01032, Development of Collective Collaboration Intelligence Framework for Internet of Autonomous Things).

References

Ahn, S., Kim, J., Lee, H., and Shin, J. Guiding deep molecular optimization with genetic exploration. *Advances in neural information processing systems*, 33:12008–12021, 2020a.

Ahn, S., Seo, Y., and Shin, J. Learning what to defer for maximum independent sets. In *Proceedings of the 37th International Conference on Machine Learning*, PMLR, pp. 134–144, 2020b.

Baptista, R. and Poloczek, M. Bayesian optimization of combinatorial structures. In *International Conference on Machine Learning*, pp. 462–471. PMLR, 2018.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., and Bengio, S. Neural combinatorial optimization with reinforcement learning. In *5th International Conference on Learning Representations, Workshop Track Proceedings*, 2017.

Bengio, E., Jain, M., Korablyov, M., Precup, D., and Bengio, Y. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394, 2021.

Bengio, Y., Lahlou, S., Deleu, T., Hu, E. J., Tiwari, M., and Bengio, E. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.

Berto, F., Hua, C., Park, J., Kim, M., Kim, H., Son, J., Kim, H., Kim, J., and Park, J. RL4CO: an extensive reinforcement learning for combinatorial optimization benchmark. *arXiv preprint arXiv:2306.17100*, 2023. URL https://github.com/kaist-silab/rl4co.

Bickerton, G. R., Paolini, G. V., Besnard, J., Muresan, S., and Hopkins, A. L. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90–98, 2012.

Bliek, L., Guijt, A., Karlsson, R., Verwer, S., and de Weerdt, M. Benchmarking surrogate-based optimisation algorithms on expensive black-box functions. *Applied Soft Computing*, 147:110744, 2023.

Chen, X. and Tian, Y. Learning to perform local rewriting for combinatorial optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

Choo, J., Kwon, Y.-D., Kim, J., Jae, J., Hottung, A., Tierney, K., and Gwon, Y. Simulation-guided beam search for neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 35:8760–8772, 2022.

Cohen, T. and Welling, M. Group equivariant convolutional networks. In *International conference on machine learning*, pp. 2990–2999. PMLR, 2016.

Deac, A., Weber, T., and Papamakarios, G. Equivariant MuZero. *arXiv preprint arXiv:2302.04798*, 2023.

Deshwal, A., Belakaria, S., Doppa, J. R., and Kim, D. H. Bayesian optimization over permutation spaces. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 6515–6523, 2022.

D'Oro, P., Schwarzer, M., Nikishin, E., Bacon, P.-L., Bellemare, M. G., and Courville, A. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.

Duan, H., Vaezipoor, P., Paulus, M. B., Ruan, Y., and Maddison, C. Augment with care: Contrastive learning for combinatorial problems. In *International Conference on Machine Learning*, pp. 5627–5642. PMLR, 2022.

Falet, J.-P. R., Lee, H. B., Malkin, N., Sun, C., Secrieru, D., Zhang, D., Lajoie, G., and Bengio, Y. Delta-AI: Local objectives for amortized inference in sparse graphical models. In *The Twelfth International Conference on Learning Representations*, 2024.

Fan, L., Wang, G., Huang, D.-A., Yu, Z., Fei-Fei, L., Zhu, Y., and Anandkumar, A. SECANT: Self-expert cloning for zero-shot generalization of visual policies. In *International Conference on Machine Learning*, pp. 3088–3099. PMLR, 2021.

Fedus, W., Ramachandran, P., Agarwal, R., Bengio, Y., Larochelle, H., Rowland, M., and Dabney, W. Revisiting fundamentals of experience replay. In *International Conference on Machine Learning*, pp. 3061–3071. PMLR, 2020.

Feng, S. Y., Gangal, V., Wei, J., Chandar, S., Vosoughi, S., Mitamura, T., and Hovy, E. A survey of data augmentation approaches for nlp. *arXiv preprint arXiv:2105.03075*, 2021.

Gao, W., Fu, T., Sun, J., and Coley, C. Sample efficiency matters: a benchmark for practical molecular optimization. *Advances in Neural Information Processing Systems*, 35:21342–21357, 2022.

Garg, S. and Ramakrishnan, G. BAE: BERT-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6174–6181, 2020.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

Hottung, A. and Tierney, K. Neural large neighborhood search for the capacitated vehicle routing problem. In *24th European Conference on Artificial Intelligence (ECAI 2020)*, 2020.

Hottung, A., Kwon, Y.-D., and Tierney, K. Efficient active search for combinatorial optimization problems. In *International Conference on Learning Representations*, 2022.

Irurozki, E. and López-Ibáñez, M. Unbalanced mallows models for optimizing expensive black-box permutation problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 225–233, 2021.

Jain, M., Bengio, E., Hernandez-Garcia, A., Rector-Brooks, J., Dossou, B. F., Ekbote, C. A., Fu, J., Zhang, T., Kilgour, M., Zhang, D., et al. Biological sequence design with gflownets. In *International Conference on Machine Learning*, pp. 9786–9801. PMLR, 2022.

Jang, H., Kim, M., and Ahn, S. Learning energy decompositions for partial inference in GFlownets. In *The Twelfth International Conference on Learning Representations*, 2024.

Jensen, J. H. A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space. *Chemical science*, 10(12):3567–3572, 2019.

Jiang, Y., Cao, Z., and Zhang, J. Solving 3d bin packing problem via multimodal deep reinforcement learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1548–1550, 2021.

Jiang, Y., Cao, Z., Wu, Y., Song, W., and Zhang, J. Ensemble-based deep reinforcement learning for vehicle routing problems under distribution shift. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Jun, H., Cho, J., Lee, K., Son, H.-Y., Kim, K., Jin, H., and Kim, K. Hbm (high bandwidth memory) dram technology and architecture. In *2017 IEEE International Memory Workshop (IMW)*, pp. 1–4. IEEE, 2017.

Khalil, E., Dai, H., Zhang, Y., Dilkina, B., and Song, L. Learning combinatorial optimization algorithms over graphs. *Advances in Neural Information Processing Systems*, 30, 2017.

Kielak, K. P. Do recent advancements in model-based deep reinforcement learning really improve data efficiency? 2019.

Kim, H., Park, H., Kim, M., Choi, S., Kim, J., Park, J., Kim, S., Kim, S., and Kim, J. Deep reinforcement learning framework for optimal decoupling capacitor placement on general pdn with an arbitrary probing port. In *2021 IEEE 30th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, pp. 1–3, 2021a. doi: 10.1109/EPEPS51341.2021.9609194.

Kim, H., Kim, M., Berto, F., Kim, J., and Park, J. DevFormer: A symmetric transformer for context-aware device placement. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 16541–16566. PMLR, 23–29 Jul 2023.

Kim, J., Kang, D., Ahn, S., and Shin, J. What makes better augmentation strategies? augment difficult but not too different. In *International Conference on Learning Representations*, 2021b.

Kim, M., Park, J., and Park, J. Learning to CROSS exchange to solve min-max vehicle routing problems. In *The Eleventh International Conference on Learning Representations*, 2022a.

Kim, M., Park, J., and Park, J. Sym-NCO: Leveraging symmetricity for neural combinatorial optimization. In *Advances in Neural Information Processing Systems*, 2022b.

Kim, M., Choi, S., Son, J., Kim, H., Park, J., and Bengio, Y. Ant colony sampling with GFlowNets for combinatorial optimization. *arXiv preprint arXiv:2403.07041*, 2024a.

Kim, M., Ko, J., Yun, T., Zhang, D., Pan, L., Kim, W., Park, J., Bengio, E., and Bengio, Y. Learning to scale logits for temperature-conditional GFlowNets. In *International Conference on Machine Learning*, 2024b.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6114.

Koo, K., Luevano, G. R., Wang, T., Özbayat, S., Michalka, T., and Drewniak, J. L. Fast algorithm for minimizing the number of decap in power distribution networks. *IEEE Transactions on Electromagnetic Compatibility*, 60(3): 725–732, 2017.

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

Korovina, K., Xu, S., Kandasamy, K., Neiswanger, W., Poczos, B., Schneider, J., and Xing, E. ChemBO: Bayesian optimization of small organic molecules with synthesizable recommendations. In *International Conference on Artificial Intelligence and Statistics*, pp. 3393–3403. PMLR, 2020.

Krenn, M., Häse, F., Nigam, A., Friederich, P., and Aspuru-Guzik, A. Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation. *Machine Learning: Science and Technology*, 1(4):045024, 2020.

Kwon, Y.-D., Choo, J., Kim, B., Yoon, I., Gwon, Y., and Min, S. POMO: Policy optimization with multiple optima for reinforcement learning. *Advances in Neural Information Processing Systems*, 33:21188–21198, 2020.

Kwon, Y.-D., Choo, J., Yoon, I., Park, M., Park, D., and Gwon, Y. Matrix encoding networks for neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 34:5138–5149, 2021.

Laskin, M., Lee, K., Stooke, A., Pinto, L., Abbeel, P., and Srinivas, A. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33: 19884–19895, 2020.

Li, J.-Y., Zhan, Z.-H., and Zhang, J. Evolutionary computation for expensive optimization: A survey. *Machine Intelligence Research*, 19(1):3–23, 2022.

Li, S., Yan, Z., and Wu, C. Learning to delegate for large-scale vehicle routing. *Advances in Neural Information Processing Systems*, 34:26198–26211, 2021.

Li, Y., Zhang, L., and Liu, Z. Multi-objective de novo drug design with conditional graph generative model. *Journal of cheminformatics*, 10:1–24, 2018.

Lindauer, M., Eggensperger, K., Feurer, M., Biedenkapp, A., Deng, D., Benjamins, C., Ruhkopf, T., Sass, R., and Hutter, F. Smac3: A versatile bayesian optimization package for hyperparameter optimization. *The Journal of Machine Learning Research*, 23(1):2475–2483, 2022.

Ma, G., Bengio, E., Bengio, Y., and Zhang, D. Baking symmetry into GFlowNets. In *NeurIPS 2023 AI for Science Workshop*, 2023a.

Ma, Y., Hao, X., Hao, J., Lu, J., Liu, X., Xialiang, T., Yuan, M., Li, Z., Tang, J., and Meng, Z. A hierarchical reinforcement learning based optimization framework for large-scale dynamic pickup and delivery problems. *Advances in Neural Information Processing Systems*, 34: 23609–23620, 2021.

Ma, Y., Cao, Z., and Chee, Y. M. Learning to search feasible and infeasible regions of routing problems with flexible

neural k-opt. In *Advances in Neural Information Processing Systems*, volume 36, pp. 49555–49578, 2023b.

Madan, K., Rector-Brooks, J., Korablyov, M., Bengio, E., Jain, M., Nica, A. C., Bosc, T., Bengio, Y., and Malkin, N. Learning GFlowNets from partial episodes for improved convergence and stability. In *International Conference on Machine Learning*, pp. 23467–23483. PMLR, 2023.

Malkin, N., Jain, M., Bengio, E., Sun, C., and Bengio, Y. Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35:5955–5967, 2022.

Mondal, A. K., Nair, P., and Siddiqi, K. Group equivariant deep reinforcement learning. *arXiv preprint arXiv:2007.03437*, 2020.

Mondal, A. K., Jain, V., Siddiqi, K., and Ravanbakhsh, S. EqR: Equivariant representations for data-efficient reinforcement learning. In *International Conference on Machine Learning*, pp. 15908–15926. PMLR, 2022.

Moss, H., Leslie, D., Beck, D., Gonzalez, J., and Rayson, P. BOSS: Bayesian optimization over string spaces. *Advances in neural information processing systems*, 33: 15476–15486, 2020.

Nikishin, E., Schwarzer, M., D'Oro, P., Bacon, P.-L., and Courville, A. The primacy bias in deep reinforcement learning. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 16828–16847. PMLR, 2022.

Oh, C., Tomczak, J., Gavves, E., and Welling, M. Combinatorial bayesian optimization using the graph cartesian product. *Advances in Neural Information Processing Systems*, 32, 2019.

Olivecrona, M., Blaschke, T., Engkvist, O., and Chen, H. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):1–14, 2017.

Pan, L., Zhang, D., Courville, A., Huang, L., and Bengio, Y. Generative augmented flow networks. In *The Eleventh International Conference on Learning Representations*, 2022.

Pan, L., Malkin, N., Zhang, D., and Bengio, Y. Better training of GFlowNets with local credit and incomplete trajectories. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 26878–26890. PMLR, 2023.

Park, J., Kwon, C., and Park, J. Learn to solve the min-max multiple traveling salesmen problem with reinforcement learning. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pp. 878–886, 2023.

Qiu, R., Sun, Z., and Yang, Y. DIMES: A differentiable meta solver for combinatorial optimization problems. In *Advances in Neural Information Processing Systems*, volume 35, pp. 25531–25546, 2022.

Rector-Brooks, J., Madan, K., Jain, M., Korablyov, M., Liu, C.-H., Chandar, S., Malkin, N., and Bengio, Y. Thompson sampling for improved exploration in GFlowNets. *arXiv preprint arXiv:2306.17693*, 2023.

Ren, S., Zhang, J., Li, L., Sun, X., and Zhou, J. Text autoaugment: Learning compositional augmentation policy for text classification. *arXiv preprint arXiv:2109.00523*, 2021.

Satorras, V. G., Hoogeboom, E., and Welling, M. $E(n)$ equivariant graph neural networks. In *International Conference on Machine Learning*, pp. 9323–9332. PMLR, 2021.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shen, D., Zheng, M., Shen, Y., Qu, Y., and Chen, W. A simple but tough-to-beat data augmentation approach for natural language understanding and generation. *arXiv preprint arXiv:2009.13818*, 2020.

Shen, M. W., Bengio, E., Hajiramezanali, E., Loukas, A., Cho, K., and Biancalani, T. Towards understanding and improving GFlowNet training. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Smith, L., Kostrikov, I., and Levine, S. A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. *arXiv preprint arXiv:2208.07860*, 2022.

Son, J., Kim, M., Kim, H., and Park, J. Meta-SAGE: Scale meta-learning scheduled adaptation with guided exploration for mitigating scale shift on combinatorial optimization. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 32194–32210. PMLR, 2023.

Son, J., Kim, M., Choi, S., Kim, H., and Park, J. Equity-Transformer: Solving NP-hard min-max routing problems as sequential generation with equity context. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 20265–20273, 2024.

Tripp, A., Simm, G. N., and Hernández-Lobato, J. M. A fresh look at de novo molecular design benchmarks. In *NeurIPS 2021 AI for Science Workshop*, 2021.

Van der Pol, E., Worrall, D., van Hoof, H., Oliehoek, F., and Welling, M. MDP homomorphic networks: Group symmetries in reinforcement learning. *Advances in Neural Information Processing Systems*, 33:4199–4210, 2020.

Vinyals, O., Fortunato, M., and Jaitly, N. Pointer networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

Wang, D. and Walters, R. $SO(2)$-equivariant reinforcement learning. In *International Conference on Learning Representations*, 2022.

Wang, X., Pham, H., Dai, Z., and Neubig, G. SwitchOut: an efficient data augmentation algorithm for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 856–861, 2018.

Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., and de Freitas, N. Sample efficient actor-critic with experience replay. In *International Conference on Learning Representations*, 2017.

Weiler, M. and Cesa, G. General $E(2)$-Equivariant steerable CNNs. *Advances in Neural Information Processing Systems*, 32, 2019.

Weininger, D. 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.

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

Xin, L., Song, W., Cao, Z., and Zhang, J. NeuroLKH: Combining deep learning model with lin-kernighan-helsgaun heuristic for solving the traveling salesman problem. *Advances in Neural Information Processing Systems*, 34: 7472–7483, 2021b.

Yao, S., Lin, X., Wang, Z., and Zhang, Q. Data-efficient supervised learning is powerful for neural combinatorial optimization. 2022.

Yarats, D., Fergus, R., and Kostrikov, I. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *9th International Conference on Learning Representations*, 2021.

Ye, H., Wang, J., Cao, Z., Liang, H., and Li, Y. DeepACO: Neural-enhanced ant systems for combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 36, pp. 43706–43728, 2023.

Zhang, C., Song, W., Cao, Z., Zhang, J., Tan, P. S., and Chi, X. Learning to dispatch for job shop scheduling via deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1621–1632, 2020.

Zhang, D., Dai, H., Malkin, N., Courville, A., Bengio, Y., and Pan, L. Let the flows tell: Solving graph combinatorial problems with gflownets. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

A. Proof for Theoretical Analysis in Section 3.2

A.1. Proof for Theorem 4.1

Consider $\pi_\theta(\tau)$ as the distribution over the trajectory. Let $\mathcal{T}_x$ denote the space of trajectories associated with the solution $\boldsymbol{x}$.

$$\begin{aligned}
\mathcal{H}(\pi_\theta(\tau)) &= -\sum_{\tau\in\mathcal{T}} \pi_\theta(\tau)\log\pi_\theta(\tau) \\
&= -\sum_{\boldsymbol{x}\in\mathcal{X}}\sum_{\tau_{\to x}\in\mathcal{T}_x} \pi_\theta(\tau)\log\pi_\theta(\tau) \\
&= -\sum_{\boldsymbol{x}\in\mathcal{X}}\sum_{\tau_{\to x}\in\mathcal{T}_x} p(\tau_{\to x}|\boldsymbol{x})p(\boldsymbol{x})\left(\log p(\tau_{\to x}|\boldsymbol{x}) + \log p(\boldsymbol{x})\right) \\
&= \mathcal{H}(p(\boldsymbol{x})) + \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\mathcal{H}(p(\tau_{\to x}|\boldsymbol{x})) \\
&\le \mathcal{H}(p(\boldsymbol{x})) + \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}\mathcal{H}(U(\tau_{\to x}|\boldsymbol{x})),
\end{aligned}$$

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

A.2. Extreme Case of Inverse Temperature in Boltzmann Distribution

Proposition A.1. *Consider $p(\tau)$ to be the Boltzmann distribution of energy $\mathcal{E}(\tau)$ with inverse temperature β given by:*

$$p(\tau) \propto e^{-\beta\mathcal{E}(\tau)}.$$

The following two statements are satisfied:

Statement 1 *If $\beta \to \infty$, then $p(\tau) = \delta_{\tau,\text{argmin}\,\mathcal{E}(\tau)}$, where $\delta_{i,j}$ is the Kronecker delta function.*

Statement 2 *If $\beta = 0$, then $p(\tau)$ is a uniform distribution.*

Proof of Statement 1. Consider the limit as $\beta \to \infty$. The Boltzmann distribution becomes:

$$\lim_{\beta\to\infty} p(\tau) \propto \lim_{\beta\to\infty} e^{-\beta\mathcal{E}(\tau)}.$$

For any τ such that $\mathcal{E}(\tau) > \min_{\tau'}\mathcal{E}(\tau')$, the exponential term $e^{-\beta\mathcal{E}(\tau)}$ approaches zero as β becomes very large. To see this, let $\epsilon = \mathcal{E}(\tau) - \min_{\tau'}\mathcal{E}(\tau') > 0$. Then, we have:

$$\lim_{\beta\to\infty} e^{-\beta\mathcal{E}(\tau)} = \lim_{\beta\to\infty} e^{-\beta(\min_{\tau'}\mathcal{E}(\tau')+\epsilon)} = 0.$$

Now, for the case where $\tau = \text{argmin}\,\mathcal{E}(\tau)$, the exponential term $e^{-\beta\mathcal{E}(\tau)}$ becomes 1, as β does not affect the term when τ minimizes the energy.

Thus, in the limit $\beta \to \infty$, we have:

$$\lim_{\beta\to\infty} p(\tau) \propto \begin{cases} 1, & \text{if } \tau = \text{argmin}\,\mathcal{E}(\tau), \\ 0, & \text{otherwise.} \end{cases}$$

Now, using the definition of the Kronecker delta function $\delta_{i,j}$:

$$\lim_{\beta\to\infty} p(\tau) \propto \delta_{\tau,\text{argmin}\,\mathcal{E}(\tau)}.$$

Thus, Statement 1 holds. □

Proof of Statement 2. For $\beta = 0$, the Boltzmann distribution becomes:

$$p(\tau) \propto e^{-0 \cdot \mathcal{E}(\tau)} = 1.$$

Therefore, $p(\tau)$ is a constant, and the distribution is uniform, satisfying Statement 2. □

Let the energy $\mathcal{E}(\tau)$ be a log-likelihood of training policy: $\mathcal{E}(\tau) = \log \pi_\theta(\tau)$. Then, statement 1 shows that our important sampling transformation becomes the adversarial transformation. Also, statement 2 shows that our important sampling transformation becomes the maximum entropy (i.e., uniform) transformation.

B. Implementation details

B.1. Pseudo-code for REINFORCE with SRT

In this section, we provide a pseudo-code in case of employing REINFORCE as a base DRL method in Step A.

Algorithm 1 REINFORCE with SRT

Input: objective function f, batch size B, SRT sample width L, a scale coefficient α
Initialize the number of reward calls $k \leftarrow 0$,
while $k \leq K$ **do**
  $s_1^i \leftarrow$ `GetInitialState()`, $\forall i \in \{1, \ldots, B\}$
  $\tau \sim \pi_\theta(\cdot|s_1) = \prod_{t=1}^{T} \pi_\theta(s_{t+1}|s_t)$
  ▷ REWARD EVALUATION
  $R(x^i) \leftarrow f(x^i), \quad \forall i \in \{1, \ldots, B\}$
  $k \leftarrow k + B$
  ▷ STEP A. REWARD-MAXIMIZING TRAINING
  $\mathcal{L}_{\text{RM}} \leftarrow \sum_{i=1}^{B} (R(x^i) - b) \log \pi_\theta(\cdot)$
  $\theta \leftarrow \text{Adam}(\theta, \nabla \mathcal{L}_{\text{RM}})$
  ▷ STEP B. SYMMETRIC REPLAY TRAINING
  $\tau^i \leftarrow$ `GreedyRollout()`, $\forall i \in \{1, \ldots, B\}$
  Symmetric transform τ, i.e., $\tau^{i,1}, \ldots \tau^{i,L} \sim p(\tau_{\to x}|x^i), \quad \forall i \in \{1, \ldots, B\}$
  $\mathcal{L}_{\text{SRT}} \leftarrow \alpha \sum_{i=1}^{B} \frac{1}{L} \sum_{l=1}^{L} \log \pi_\theta(\tau^{i,l}|s_1^i)$
  $\theta \leftarrow \text{Adam}(\theta, \nabla \mathcal{L}_{\text{SRT}})$
end while

There are various ways to collect high-reward samples in Step B. For example, `GreedyRollout` can be replaced with other strategies, such as reward-prioritized sampling or collecting Top-k samples.

B.2. Proximal policy optimization (PPO)

We use AM architecture (Kool et al., 2018) on TSP and Devformer architecture (Kim et al., 2023) on DPP for parameterizing compositional policy $\pi(\boldsymbol{x}|s_1) = \prod_{t=1}^{N} \pi(a_t|s_t)$. Then, we implement based on the following equation as follows:

$$\mathcal{L}(\boldsymbol{x}; s_1) = \min \left[A(\boldsymbol{x}; s_1) \frac{\pi(\boldsymbol{x}|s_1)}{\pi_{\text{old}}(\boldsymbol{x}|s_1)}, A(\boldsymbol{x}; s_1) \text{clip} \left(\frac{\pi(\boldsymbol{x}|s_1)}{\pi_{\text{old}}(\boldsymbol{x}|s_1)}, 1 - \epsilon, 1 + \epsilon \right) \right],$$
$$A(\boldsymbol{x}; s_1) = R(\boldsymbol{x}; s_1) - V(s_1),$$

where R stands for reward function and V stands for value function. Since we implement PPO on compositional MDP setting, we train value function in the context of s_1 by following actor-critic implementation of Kool et al. (2018).

Hyperparameters. We systematically investigate a range of hyperparameter combinations involving different baselines ([rollout, critic]), various values for clipping epsilon ([0.1, 0.2, 0.3]), and numbers of inner loops ([5, 10, 20]). Our observations reveal that the critic baseline consistently enhances training stability across all tasks, leading to reduced variation when modifying the training seeds. The best configurations for each task are provided in Table 4.

Table 4. Hyperparameter configurations for PPO.

	TSP	Chip-package PDN	HBM PDN
Baseline	critic	critic	critic
Eps. clip	0.2	0.1	0.2
Number of inner loops k	5	20	10

B.3. Generative flow network (GFlowNet)

Similar to the PPO implementation, we employ the Attention Model (AM) architecture (Kool et al., 2018) on the Traveling Salesman Problem (TSP), and the DevFormer architecture (Kim et al., 2023) on DPP, for parameterizing the compositional forward policy $P_F(\tau|s_1) = \prod_{t=1}^{N} P_F(a_t|s_t)$. Subsequently, we configure the backward policy P_B as a uniform distribution for all possible parent nodes, following the methodology outlined in (Malkin et al., 2022). Lastly, we parameterize $Z(s_1)$ using a two-layer perceptron with ReLU activation functions, where the number of hidden units matches the embedding dimension of the AM or DevFormer. This two-layer perceptron takes input from the mean of the encoded embedding vector obtained from the encoder of the AM or DevFormer and produces a scalar value to estimate the partition function.

To train the GFlowNet model, we use trajectory balance loss introduced in Malkin et al. (2022) as follows:

$$\mathcal{L}(\tau; s_1) = \left(\log \left(\frac{Z(s_1) P_F(\tau|s_1)}{e^{-\beta E(\boldsymbol{x}; s_1)} P_B(\tau|s_1)} \right) \right)^2 \tag{3}$$

The trajectory τ includes a terminal state represented as $\boldsymbol{x}$. Subsequently, we employ an on-policy optimization method to minimize Equation (3), with trajectories τ sampled from the training policy P_F. In this context, $E(\boldsymbol{x}; s_1)$ represents the energy, which is essentially the negative counterpart of the reward $R(\boldsymbol{x}; s_1)$. The hyperparameter β plays the role of temperature adjustment in this process.

Hyperparameters. We explore a spectrum of hyperparameter combinations, varying β ([5, 10, 20]) and numbers of inner loops ([2, 5, 10]). The best configurations for each task are provided in Table 5.

Table 5. Hyperparameter configurations for GFlowNet.

	TSP	Chip-package PDN	HBM PDN
β	20	10	10
Number of inner loops k	10	2	2

C. Experimental details

C.1. Traveling salesman problems (TSP)

Since we employ the AM architecture, we use the same hyperparameters for the model architecture and training parameters except for the batch and epoch data sizes.[3] Initially, the Attention Model (AM) employed a batch size of 512 and an epoch data size of 1,280,000. Notably, the evaluation of the greedy rollout baseline was conducted every epoch. When the number of available training samples is constrained, utilizing a smaller batch size and epoch data size becomes advantageous. Consequently, we adjusted these parameters to be 100 for batch size and 10,000 for epoch data size. In symmetric replay training (Step B), the scale coefficients are meticulously set to scale the SRT loss. As a rough guideline, we establish a coefficient that renders the SRT loss approximately 10 to 100 times smaller than the RL loss. Additionally, for the number of symmetric transformations (L in Equation (1)) is set as the number of inner loops. See Table 6 in details.

Table 6. Scale coefficient and the number of symmetric transformations in TSP.

	A2C	PG-Rollout	PPO	GFlowNet
Scale coefficient	0.001	0.001	0.00001	0.1
L	1	1	5 ($= k$)	10 ($= k$)

C.2. Decap placement problems (DPP)

Similar to the experiments on TSP, we follow the setting of DevFormer.[4] We set the batch size as 100 and epoch data size as 600. Note that the maximum number of reward calls is set 15K,, a considerably smaller limit compared to TSP. Regarding the scale coefficient and the number of symmetric transformations, we maintain consistency with the principles applied in the TSP experiments as follows:

Table 7. Scale coefficient and the number of symmetric transformations in DPP tasks.

	A2C	PG-Rollout	PPO	GFlowNet
Scale coefficient	0.01	0.01	0.01	0.1
L	1	1	20 ($= k$)	2 ($= k$)

C.3. Practical molecular optimization (PMO)

We basically follow the experimental setting (e.g., batch size) in the practical molecular optimization (PMO) benchmark.[5] In symmetric replay training, we utilize reward-prioritized sampling for the online buffer, which contains molecules generated during online learning. For the REINVENT, where the replay buffer is already incorporated, we set the number of replaying samples equal to the replay buffer size, i.e., 24, and the scale coefficient to 0.001. Regarding the GFlowNet method, we configure the number of replaying samples to match the batch size of 64. Furthermore, we set the scale coefficient to 1.0, given that the RL loss in GFlowNet is much higher compared to REINVENT.

[3]AM: https://github.com/wouterkool/attention-learn-to-route
[4]DevFormer: https://github.com/kaist-silab/devformer
[5]Practical molecular optimization: https://github.com/wenhao-gao/mol_opt

D. Additional Experiments

D.1. Ablation study for the number of symmetric trajectories in SRT

We conducted the ablation study for varying sample width in replay training. 'Non-symmetric' denotes the replay training without symmetric transformation; thus, increased the number of trajectories gives duplicated samples. Figure 8 shows that the symmetric replay training consistently gives better performance and robust to the choice of L.

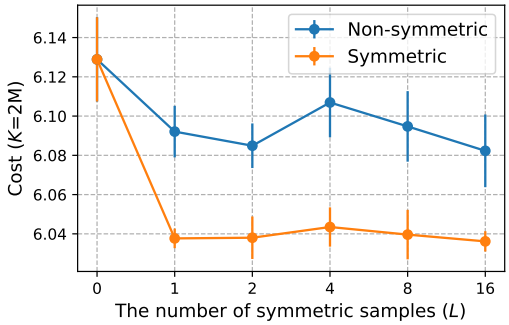

Figure 8. Ablation study for the sample width in SRT

D.2. Comparison with Experience Replay on TSP50

We conduct comparative experiments with experience replay and ours using A2C on TSP50. This requires importance sampling weight as follows:

$$\mathbb{E}_{\tau\sim\pi(\cdot|s_1)}R(\boldsymbol{x}) = \mathbb{E}_{\tau\sim q(\cdot|s_1)}\left[\frac{\pi(\tau|s_1)}{q(\tau|s_1)}R(\boldsymbol{x})\right] = \mathbb{E}_{\tau\sim q(\cdot|s_1)}\left[\prod_{t=1}^{T}\frac{\pi(s_{t+1}|s_t)}{q(s_{t+1}|s_t)}R(x)\right],$$

where $\pi(\cdot|s_1)$ and $q(\cdot|s_1)$ are the training policy and behavior policy, respectively, and x is a solution (i.e., the terminal state).

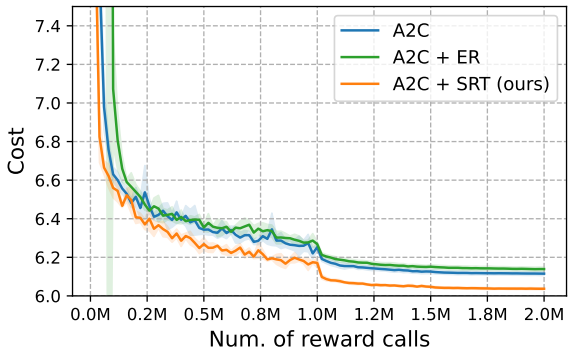

Figure 9. Comparative experiments with experience replay and SRT. The average validation costs over the computation budget are measured.

The results show that the existing experience replay method can suffer from high variance because of importance sampling, leading to the degradation of performance. Note that the importance weights are required even without symmetric transformation when using on-policy training, like A2C. On the other hand, ours does not require the importance sampling weight since SRT uses imitation learning loss.

D.3. Comparison with Data Augmentation on TSP50

Without SRT, symmetric trajectories can be utilized like data augmentation. In this experiment, we train the policy both using sampled trajectories and symmetric (i.e., augmented) trajectories. When on-policy methods like A2C, the importance weights are required for symmetric trajectories similar to the previous section. The policy is updated only using the reward-maximizing loss $\mathcal{L}_{\text{RM}}(\tau) + \mathcal{L}_{\text{RM}}(\tau')$, where $\tau' \sim p(\hat{\tau}_{\rightarrow x}|x)$. Here, $\hat{\tau}$ is obtained greedy-rollout; see Algorithm 1. The experiments are conducted with the A2C method on TSP with $N = 50$.

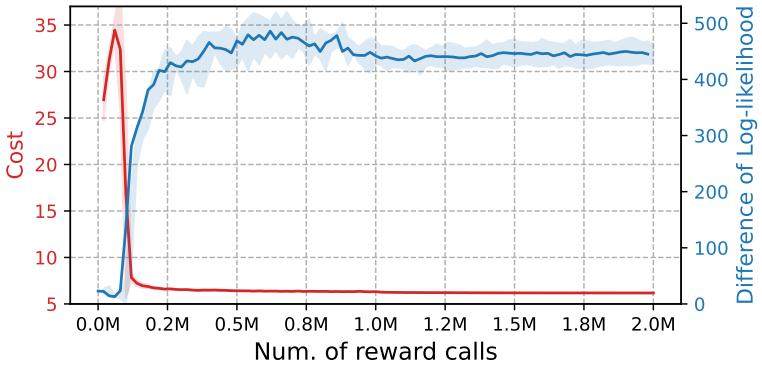

Figure 10. Optimization curve (*red*) and difference of log-likelihood between symmetric trajectories (*blue*) over training steps with reward-maximizing training of symmetric trajectories.

Table 8. The average cost and difference of log-likelihood on TSP50 with reward-maximizing training of symmetric trajectories

	100K	2M
Avg. Cost (↓)	16.851 ± 11.162	6.179 ± 0.0029
Log-likelihood Difference (↓)	23.647 ± 22.331	445.044 ± 19.053

As depicted in Figure 10 and Table 8, reward-maximizing training tends to seek one of minimizing cost or reducing differences of symmetric trajectories, not both. In the early stage of training, the cost tends to go higher because the sampled and augmented trajectories provide dis-aligned information, as the clearly increased average costs compared to the original A2C method (16.851 >> 6.780). From a certain point, the cost is decreased with a significant increase in the difference of log-likelihood. This implies that the symmetric nature of CO needs to be utilized carefully so as not to harm the performance of DRL methods, especially when on-policy learning is used.

D.4. Additional results on hardware design optimization

To verify the benefit of SRT, we also report the overall runtime (for 15K evaluations) of the hardware design optimization tasks as follows.

Table 9. Wall-clock time of whole training in hardware design optimization tasks

	Chip-package PDN	HBM PDN
A2C	4,204.50 (±342.90)	33,289.75 (±682.85)
A2C + SRT	4,265.25 (±425.38)	33,294.75 (±1,722.59)
PG-Rollout	3,747.00 (±516.97)	34,251.25 (±573.18)
PG-Rollout + SRT	3,795.25 (±803.31)	30,351.25 (±1,150.44)
PPO	3,884.25 (±81.15)	32,148.75 (±623.47)
PPO + SRT	4,700.00 (±58.26)	32,479.25 (±168.71)
GFlowNet	3,528.75 (±26.66)	35,968.00 (±3,942.48)
GFlowNet + SRT	3,528.25 (±17.80)	35,867.75 (±5,307.87)

It is worth emphasizing that SRT is beneficial when the reward evaluation costs are more expensive than the additional costs from SRT. Furthermore, SRT allows GPU batch processing, resulting much smaller wall-clock runtime compared to reward evaluation with simulation. The results show that SRT effectively enhances performance by achieving significantly lower costs while maintaining similar or slightly increased runtime, which is negligible compared to the simulation time. We will include this result in the manuscript.

D.5. Statistical analysis on molecular optimization

We perform a t-test to compare the average AUC10 scores, setting the null hypothesis as $\mu_{Base} = \mu_{SRT}$. The resulting p-values are 0.0045 and 0.0015 for REINVENT and GFlowNets, respectively. Thus, the statistical analysis indicates that SRT enhances sample efficiency significantly for both REINVENT and GFlowNets.

Table 10. The results of t-statistical test

	REINVENT	REINVENT + SRT	GFlowNet	GFlowNet + SRT
Mean	0.615	0.633	0.431	0.489
Variance	1.32E-05	7.36E-05	8.98E-06	7.83063E-05
Observations	5	5	5	5
Hypothesized Mean Diff.	0		0	
t Stat	-4.1332		-6.4356	
P($T \leq t$) one-tail	**0.0045**		**0.0015**	
t Critical one-tail	2.0150		2.1318	

D.6. Additional results on molecular optimization

In this subsection, we provide the additional results of the black-box optimization method in the PMO benchmark. Gaussian process Bayesian optimization (GPBO; Tripp et al., 2021) and Graph Genetic Algorithm (Graph GA; Jensen, 2019) are included. Note that GPBO employs Graph GA when optimizing the GP acquisition function. Though Graph GA demonstrates powerful performance, designing operators, such as crossover and mutation, greatly affects performance (Li et al., 2022) Thus, careful algorithm design, which requires specific domain knowledge, is necessitated whenever there is a change in tasks. The results show that SRT outperforms other black-box optimization methods by improving the on-policy RL method, REINVENT.

Additionally, we apply our method on REINVENT with SMILES, the simplified molecular-input line-entry system (Weininger, 1988), which is another string-based representation. It is worth mentioning that REINVENT-SMILES gives the best performance in practical molecular optimization benchmark (Gao et al., 2022). Noticeably, the results in Table 11 demonstrate that our achieves improved performances on 16 out of 23 oracles compared to REINVENT-SMILES.

Table 11. Experimental results on sample efficient molecular optimization, including REINVENT-SMILES and REINVENT-SMILES with ours. Area under the curve of top-10 average property value (↑). Note that the improved cost via SRT is denoted in **bold**.

Oracle	GPBO	Graph GA	REINVENT SMILES	REINVENT-SMILES + SRT (ours)
albuterol_similarity	0.896 ± 0.009	0.838 ± 0.027	0.881 ± 0.016	**0.890 ± 0.020**
amlodipine_mpo	0.577 ± 0.042	0.649 ± 0.014	0.645 ± 0.018	**0.654 ± 0.034**
celecoxib_rediscovery	0.733 ± 0.026	0.682 ± 0.127	0.719 ± 0.016	**0.734 ± 0.059**
deco_hop	0.620 ± 0.008	0.601 ± 0.004	0.665 ± 0.042	0.643 ± 0.009
drd2	0.933 ± 0.014	0.968 ± 0.006	0.957 ± 0.007	**0.966 ± 0.007**
fexofenadine_mpo	0.723 ± 0.002	0.773 ± 0.014	0.780 ± 0.012	**0.787 ± 0.011**
gsk3b	0.878 ± 0.018	0.792 ± 0.092	0.881 ± 0.036	0.877 ± 0.030
isomers_c7h8n2o2	0.912 ± 0.023	0.944 ± 0.030	0.941 ± 0.012	**0.961 ± 0.008**
isomers_c9h10n2o2pf2cl	0.542 ± 0.383	0.831 ± 0.018	0.840 ± 0.026	**0.874 ± 0.021**
jnk3	0.588 ± 0.095	0.677 ± 0.120	0.777 ± 0.030	**0.813 ± 0.070**
median1	0.288 ± 0.003	0.265 ± 0.016	0.372 ± 0.014	0.364 ± 0.002
median2	0.298 ± 0.005	0.268 ± 0.013	0.282 ± 0.002	**0.289 ± 0.012**
mestranol_similarity	0.659 ± 0.108	0.550 ± 0.032	0.632 ± 0.041	**0.658 ± 0.030**
osimertinib_mpo	0.788 ± 0.003	0.818 ± 0.007	0.833 ± 0.010	**0.840 ± 0.007**
perindopril_mpo	0.495 ± 0.005	0.498 ± 0.009	0.536 ± 0.015	**0.538 ± 0.009**
qed	0.936 ± 0.001	0.939 ± 0.001	0.941 ± 0.000	0.941 ± 0.000
ranolazine_mpo	0.737 ± 0.007	0.716 ± 0.011	0.770 ± 0.005	**0.796 ± 0.008**
scaffold_hop	0.536 ± 0.007	0.506 ± 0.016	0.552 ± 0.023	**0.562 ± 0.014**
sitagliptin_mpo	0.422 ± 0.008	0.486 ± 0.007	0.466 ± 0.039	**0.483 ± 0.019**
thiothixene_rediscovery	0.565 ± 0.032	0.494 ± 0.010	0.544 ± 0.029	0.539 ± 0.027
troglitazone_rediscovery	0.417 ± 0.025	0.421 ± 0.041	0.456 ± 0.021	0.428 ± 0.026
valsartan_smarts	0.000 ± 0.000	0.000 ± 0.000	0.182 ± 0.363	0.000 ± 0.000
zaleplon_mpo	0.456 ± 0.019	0.449 ± 0.012	0.533 ± 0.009	**0.555 ± 0.015**
Average	0.609	0.616	0.660	0.661

Table 12. 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
	SRT (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
	SRT (ours)	$\mathbf{12.922 \pm 0.071}$	$\mathbf{11.721 \pm 0.093}$	$\mathbf{21.582 \pm 0.149}$	$\mathbf{18.304 \pm 0.109}$

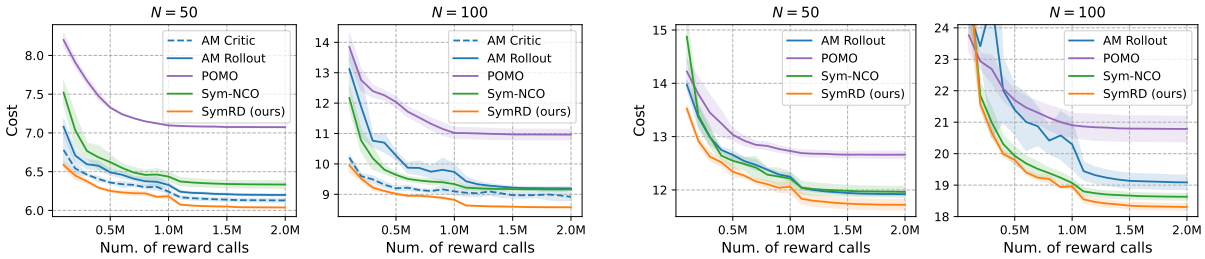

Figure 11. Validation cost over computation budget on euclidean CO problems.

D.7. Experiments on various synthetic CO problems

The experiments in this section cover various sample-efficient tasks in Euclidean and non-Euclidean combinatorial optimization. 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.

D.7.1. Euclidean CO problems

Experimental settings. We selected two representative routing tasks - the traveling salesman problem (TSP) and the capacitated vehicle routing problem (CVRP) with 50 and 100 customers. 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. For base DRL methods, we employ the best-performing DRL methods, AM for TSP and Sym-NCO for CVRP. We follow the reported hyperparameters for the model in their original paper.[6]

Results. The results in Table 12 and Figure 11 indicate that SRT 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 12, 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, SRT successfully improves sample efficiency.

[6]Sym-NCO: https://github.com/alstn12088/Sym-NCO

Table 13. 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
	SRT (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
	SRT (ours)	$\mathbf{56.104 \pm 0.125}$	$\mathbf{55.110 \pm 0.061}$	$\mathbf{96.030 \pm 0.132}$	$\mathbf{94.934 \pm 0.051}$

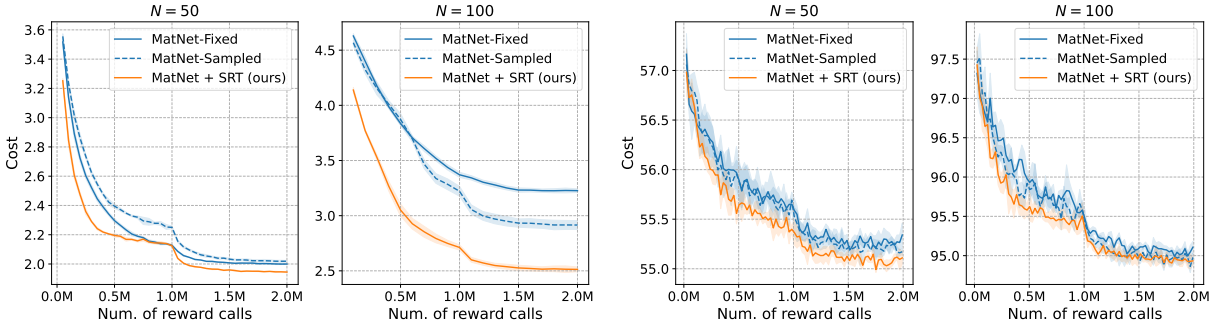

Figure 12. Validation cost over computation budget on non-Euclidean CO problems.

D.7.2. Non-Euclidean CO problems

Experimental settings. Based on the work of Kwon et al. (2021), 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. As a baseline, we employ Matrix Encoding Network (MatNet) proposed to solve non-Euclidean CO.[7] 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.

Results. The superior performance of SRT over MatNet-Fixed and MatNet-Sampled is demonstrated in both Table 13 and Figure 12. 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, SRT 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 SRT achieves 3.771 and MatNet-Sampled achieves 4.324.

[7]MatNet: https://github.com/yd-kwon/MatNet

E. Further related works

E.1. Deep reinforcement learning for combinatorial optimization

Deep reinforcement learning (DRL) has emerged as a promising methodology for solving combinatorial optimization. Especially by selecting actions sequentially, i.e., in a constructive way, DRL policies generate a combinatorial solution. This approach is beneficial to producing feasible solutions that satisfy the complex constraints of CO by restricting action space using a masking scheme (Kool et al., 2018). The foundational work of Bello et al. (2017) introduced the actor-critic method for training PointerNet (Vinyals et al., 2015) to solve TSP and the knapsack problem. Subsequently, several works were proposed to extend PointerNet into a Transformer-based model (Kool et al., 2018; Xin et al., 2021a) especially for routing problems. Building upon the success of the attention model (AM; Kool et al., 2018), Kwon et al. (2020), and Kim et al. (2022b) suggested enhanced reinforcement learning techniques by employing a precise baseline for REINFORCE based on symmetries in CO problems. On the other hand, various works have been suggested to solve broader ranges of CO problems (Khalil et al., 2017; Ahn et al., 2020b; Zhang et al., 2020; Jiang et al., 2021; Kwon et al., 2021; Kim et al., 2021a; Park et al., 2023; Zhang et al., 2023; Son et al., 2024)

Several studies have proposed to address challenges such as distributional shift and scalability (Hottung et al., 2022; Li et al., 2021; Ma et al., 2021; Choo et al., 2022; Qiu et al., 2022; Son et al., 2023; Ma et al., 2023b; Jiang et al., 2023). It is noteworthy that besides the constructive approach, there is another stream, the improvement heuristic style (Hottung & Tierney, 2020; Xin et al., 2021b; Kim et al., 2022a; Ye et al., 2023; Kim et al., 2024a), though such studies fall outside our research scope. Our research goal is to enhance the sample efficiency of constructive DRL methods for CO; the sample efficiency has been comparatively less explored in contrast to issues such as distributional shift and scalability in DRL for CO literature. This study offers an orthogonal but generally applicable approach to existing works in the field.

E.2. Equivariant Deep Learning

Equivariant DRL has also been extensively studied in recent years (Mondal et al., 2022; Van der Pol et al., 2020; Mondal et al., 2020; Wang & Walters, 2022; Deac et al., 2023). This approach reduces search space by cutting out symmetric space using equivariant representation learning, such as employing equivariant neural networks (Cohen & Welling, 2016; Weiler & Cesa, 2019; Satorras et al., 2021). Consequently, it leads to better generalization and sample efficiency. 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 available when guaranteeing equivariance is crucial.

E.3. Generative Flow Networks in Combinatorial Optimization

Genetic Flow Networks or GFlowNets (Bengio et al., 2023) are suggested to sample discrete object x from the target energy distribution, where $P(x) \propto e^{-\beta\mathcal{E}(x)}$. GFlowNets consider symmetric trajectories by modeling a sequential construction of combinatorial solutions as a flow on a directed acyclic graph, where each node corresponds to a state (a partial solution). GFlowNets have been employed to solve various CO problems, whether when the reward evaluations are expensive or not. Bengio et al. (2021) firstly proposed GFlowNets with flow matching loss, a temporal difference-like loss. The work of Bengio et al. (2021) is used to generate molecular graphs in de nove molecular optimization. Then, Jain et al. (2022) introduced a model-based GFlowNet, called GFlowNet-AL, to generate biological sequences. In addition, Zhang et al. (2023) used GFlowNet to solve CO defined on graphs, like maximum independent set (MIS) problems.

On the other hand, several works have been proposed for better GFlowNet training, by introducing new loss functions (Bengio et al., 2023; Malkin et al., 2022; Madan et al., 2023), improved exploration strategies (Pan et al., 2023; Jang et al., 2024), experience replaying (Ma et al., 2023a), and credit assignment methods (Shen et al., 2023; Pan et al., 2022; Rector-Brooks et al., 2023; Kim et al., 2024b).

E.4. Black-box Combinatorial Optimization

Building on the great success of Bayesian optimization (BO) in black-box optimization (Bliek et al., 2023; Irurozki & López-Ibáñez, 2021; Lindauer et al., 2022), several works were suggested to apply BO to combinatorial decision. Combinatorial Bayesian optimization solves a bi-level optimization where the upper problem is surrogate regression, and the lower problem is acquisition optimization. In the context of combinatorial space, the acquisition optimization is modeled as quadratic integer programming problems, which is NP-hard (Baptista & Poloczek, 2018; Deshwal et al., 2022).

To address the NP-hardness of acquisition optimization, various techniques have been proposed, including continuous relaxation Oh et al. (2019), sampling with simulated annealing (Deshwal et al., 2022), genetic algorithms (Moss et al., 2020; Tripp et al., 2021), and random walk explorer (Korovina et al., 2020). While these methods have demonstrated competitive performance in lower-dimensional tasks like neural architecture search (NAS), they often demand substantial computation time, particularly in molecular optimization tasks (Gao et al., 2022).

As mentioned in (Deshwal et al., 2022) and (Gao et al., 2022), one of the alternative approaches is to utilize a variational auto-encoder (VAE; Kingma & Welling, 2014) to map the high-dimensional combinatorial space into "compact" continuous latent space to apply BO, like (Gómez-Bombarelli et al., 2018). Though this approach has shown successful performance in molecular optimization, it also introduces variational error since VAE maximizes a lower bound of the likelihood, known as evidence lower bound (ELBO), not directly maximizes the likelihood.