CADO: Cost-Aware Diffusion Solvers for Combinatorial Optimization through RL fine-tuning

Anonymous Authors[1]

Abstract

Combinatorial Optimization (CO) problems play a pivotal role in various domains, including operational research and computer science while it have significant computational challenges. Recent advancements in Machine Learning (ML), particularly through Supervised Learning (SL) and Reinforcement Learning (RL), have shown promise in tackling these challenges. SL methods have effectively imitated high-quality solutions, while RL techniques directly optimize objectives but struggle with large-scale problems due to sparse rewards and high variance. We proposes an RL fine-tuning framework, combining SL and RL, for diffusion-based CO solvers, addressing limitations of existing methods which often ignore cost information and overlook cost variations during post-processing. Our experiments demonstrate that RL fine-tuning significantly enhances performance, surpassing traditional diffusion models and proving robust even with suboptimal training data. This approach also facilitates transfer learning across different CO problem scales, setting a new benchmark for generative model-based CO solvers.

1. Introduction

Combinatorial Optimization (CO) problems play a pivotal role in various domains, including operational research and computer science. However, the inherent complexity of these problems, such as NP-hardness, poses significant computational challenges (Karp, 1975). Traditionally, the field has been dominated by rule-based heuristics tailored to specific problems (Papadimitriou & Steiglitz, 1998). Nevertheless, recent advancements in Machine Learning (ML) have demonstrated the potential to tackle CO problems through data-driven approaches (Bengio et al., 2021). ML-based CO solvers can be broadly categorized into two: Supervised Learning (SL) methods and Reinforcement Learning (RL) techniques. The key difference between these approaches lies in the availability of a training dataset comprising solutions as labels for CO instances.

Supervised Learning (SL) methods have shown promising results in solving Combinatorial Optimization (CO) problems by imitating high-quality solutions from a training dataset (Graikos et al., 2022a; Mirhoseini et al., 2021; Kool et al., 2019a; Niu et al., 2020). Recent advancements in generative models, such as diffusion models, have demonstrated remarkable precision in generating high-dimensional outputs in image and language domains (Ho et al., 2020). These successes have also been applied to CO domains, with notable examples like DIFUSCO (Sun & Yang, 2023) and T2T (Li et al., 2023). Despite the promising results of applying diffusion models to CO problems, training without considering cost information can be particularly problematic in the CO domain, because prediction errors in generated solutions might be similar, but their costs can vary significantly. As a result, the trained model may produce undesirable outcomes that do not satisfy the true objective during inference. On the other hand, RL methods (da Costa et al., 2020; Wu et al., 2019; Kool et al., 2019b; Kwon et al., 2020; Kim et al., 2022) directly optimize the objective but pose challenges for large-scale problems due to sparse reward problem and high training variance.

Meanwhile, combining RL and SL in order to complement each other has proven highly successful in various domains such as image and texts (Ziegler et al., 2019; Deng et al., 2022; Bai et al., 2022; Clark et al., 2024; Fan et al., 2023; Black et al., 2024). These hybrid methods typically involve using (self) supervised learning to train a large generative model on large datasets and then fine-tuning it with RL to optimize the true objective. RL fine-tuning helps refine the generative model to better meet the desired objectives, making it crucial for practical applications.

Inspired by the recent success, we propose an RL fine-tuning framework for CO. Despite its success in various domains, the effectiveness of RL fine-tuning on generative models has not yet been explored in the CO domain. Figure 1

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the SPIGM workshop at ICML 2024. Do not distribute.

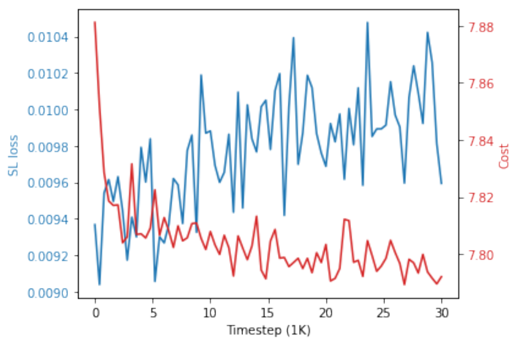

Figure 1. The learning curve during RL fine-tuning. The cost value decreases, but the SL loss increases, indicating that the SL loss is not sufficient for CO.

shows the prediction error (SL loss) between the solutions generated by the model and the optimal solution, and cost values during RL fine-tuning where the cost decreases, the SL loss increases. This indicates that training using only the SL loss may not be sufficient in CO and necessity for the cost utilization.

We apply RL fine-tuning to the diffusion-based solver, which has shown promising results in CO, and it offers additional advantages for diffusion model-based CO solvers. The necessity of feasibility in CO requires solutions to adhere to strict constraints. Sun & Yang (2023) employs post-processing decoders to transform raw solutions sampled from the diffusion model into constraint-satisfying ones, but their learning objective ignores the potential changes in the solution's cost during post-processing, which can lead to suboptimal performance. Through RL fine-tuning, the model learns to generate solutions with post-processing decoding in mind.

In our comprehensive experiments, our RL-finetuning framework demonstrates superiority in various scenarios. First, our fine-tuned diffusion model outperforms other diffusion baselines by being aware of the changes in the cost of the post-processed solutions. Second, our ideas can be applied for transfer learning across different scales of CO domains, making it easier to adapt the model to new problem instances of varying sizes without requiring additional training datasets for each size. Finally, unlike existing methods that rely heavily on high-quality training datasets, our approach shows robustness even with suboptimal training data. This is crucial for real-world applications where optimal solutions are not always available for training. These overall results suggest that our integration of cost information and the decoding process into the learning framework offers a promising improvement for generative model-based CO solvers.

2. Preliminaries and Related works

In this section, we provide the preliminary knowledge and the most related works. Additional related works are provided in Appendix A.

2.1. Problem Formulation

We define the problem and introduce the key notations related to combinatorial optimization (CO) problems. Let $\mathcal{G}$ be the set of all CO instances, and let $g \in \mathcal{G}$ denote a instance. Each instance g has an associated discrete solution space $\mathcal{X}_g := \{0,1\}^{N_g}$ and an objective function $c_g : \mathcal{X}_g \to \mathbb{R}$ for each solution $\boldsymbol{x} \in \mathcal{X}_g$ defined as:

$$c_g(\boldsymbol{x}) = \text{cost}(\boldsymbol{x}, g) + \text{valid}(\boldsymbol{x}, g). \tag{1}$$

Here, $\text{cost}(\cdot)$ represents the cost value to be optimized, while $\text{valid}(\cdot)$ is a constraint indicator function, where $\text{valid}(\boldsymbol{x}, g) = 0$ if the solution $\boldsymbol{x}$ belongs to the feasible solution space $\mathcal{F}_g \subset \mathcal{X}_g$, and $\text{valid}(\boldsymbol{x}, g) = \infty$ when $\boldsymbol{x} \notin \mathcal{F}_g$. The optimization goal is to find the optimal solution $\boldsymbol{x}_\star$ for a given instance s:

$$\boldsymbol{x}_\star^g = \arg\min_{\boldsymbol{x} \in \mathcal{X}_g} c_g(\boldsymbol{x}). \tag{2}$$

We describe two specific CO problems as examples: the Traveling Salesman Problem (TSP) and the Maximal Independent Set (MIS) problem. In the TSP, an instance g represents the coordinates of n cities to be visited. The solution $\boldsymbol{x}$ is an $n \times n$ matrix, where $\boldsymbol{x}[i, j] = 1$ if the traveler moves from city i to city j or vice versa. The total solution space is $\mathcal{X}_g = \{0,1\}^{n \times n}$, and the feasible solution space $\mathcal{F}_g \subset \mathcal{X}_g$ is the set of all feasible TSP tours that visit each city exactly once. The objective function $\text{cost}(\cdot)$ represents the total length of the given tour and should be minimized. In the MIS problem, an instance g represents a graph (V, E), where V is the vertex set and E is the edge set. The solution space $\mathcal{X}_g = \{0,1\}^V$ indicates whether each vertex $v \in V$ is included in the solution set. To satisfy the independence property, $\boldsymbol{x}$ should not contain nodes connected by edges in E. The objective function $\text{cost}(\cdot)$ represents the total number of selected nodes and should be maximized.

2.2. Supervised Learning and Reinforcement Learning in Combinatorial Optimization

As described in the introduction, most Neural Combinatorial Optimization (NCO) approaches can be categorized into two types: Supervised Learning (SL) and Reinforcement Learning (RL). In this part, we compare the learning objectives for both approaches. In SL, the solver assumes the availability of high-quality solutions $\boldsymbol{x}_g^\star$ for each training instance $g \sim \boldsymbol{P}(g)$, where $\boldsymbol{P}(g)$ is the distribution of the CO instances.

The solver's goal is to search for parameters θ that resemble a conditional distribution of the high-quality solutions $p_\theta(\boldsymbol{x}_\star^g|g) \approx \boldsymbol{P}(\boldsymbol{x}_\star^g|g)$ for a given instance $g \sim \boldsymbol{P}(g)$. Typically, generative model-based CO solvers try to maximize the likelihood using the following objective function $L(\theta)$:

$$L(\theta) = \mathbb{E}_{g\sim \boldsymbol{P}(g)}[-\log p_\theta(\boldsymbol{x}_\star^g|g)]. \tag{3}$$

One notable point is that the solver in SL just resembles the distribution of the optimal solution $\boldsymbol{x}_\star^g$ rather than considering the cost$(\boldsymbol{x}, g)$ function.

In RL, the solver does not assume the availability of the high-quality solutions $\boldsymbol{x}_\star^g$ for a given instance g. However, the solver exploits the information of the objective function $c_g(\cdot)$ during exploration and exploitation of the solutions $\boldsymbol{x}$. The solver's goal is also to learn a distribution $p_\theta(\boldsymbol{x} \mid g)$ for a given instance g that optimizes the objective function c_g as follows:

$$R(\theta) = \mathbb{E}_{g\sim \boldsymbol{P}(g),\, \boldsymbol{x}\sim p_\theta(\boldsymbol{x}|g)}[-c_g(\boldsymbol{x})]. \tag{4}$$

Although both approaches are guaranteed to find the optimal parameters under ideal conditions, when applied in practice, each has its own pros and cons. In the case of SL, a sufficiently large amount of high-quality training data is required, but due to the NP-hardness of the CO problem, it is not easy to create this data. In the case of RL, it is difficult to learn because it starts from scratch due to the limitation of the solutions worth referencing.

2.3. Diffusion model for CO

Sun & Yang (2023) propose a diffusion model-based CO solver called DIFUSCO. In CO, a diffusion model is employed to estimate the distribution of high-quality solutions for combinatorial optimization problems during the training phase (Sun & Yang, 2023; Li et al., 2023). Since the solution $\boldsymbol{x}$ is belongs to the discrete solution space $\{0,1\}^N$, the noising process $q(\mathbf{x_t}|\mathbf{x_{t-1}})$ and denoising process $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ are also done on the discrete space $\{0,1\}^N$. In this work, we followed the discrete diffusion models introduced by Austin et al. (2021a); Hoogeboom et al. (2021); Sun & Yang (2023).

The diffusion process consists of a forward noising procedure and a reverse denoising procedure. The forward process incrementally adds noise to the initial solution $\mathbf{x}_0 = \boldsymbol{x}_\star^g$, creating a sequence of latent variables $\mathbf{x_0}, \mathbf{x_1}, \ldots, \mathbf{x_T}$. Note that in CO, $\mathbf{x}_0$ follows the high-quality solutions for a given instance g, i.e., $\mathbf{x}_0 \sim \boldsymbol{P}(\boldsymbol{x}_\star^g|g)$. Furthermore, the fully noised solution $\mathbf{x}_T$ in the last timestep T becomes an N_g dimensional Bernoulli random variable with probability $\mathbf{p} = \{0.5\}^{N_g}$ and each variable is independent of each other, i.e., $\mathbf{x}_T \sim \text{Bern}(\mathbf{p} = \{0.5\}^{N_g})$. For brevity, we omit a problem instance g and denote $\boldsymbol{x}_\star^g$ as $\mathbf{x_0}$ in all formulas of the diffusion model as a convention.

The forward noising process is defined by $q(\mathbf{x_{1:T}}|\mathbf{x_0}) = \prod_{t=1}^T q(\mathbf{x_t}|\mathbf{x_{t-1}})$, where $\mathbf{x_0} \sim q(\mathbf{x_0}|g)$, and $q(\mathbf{x_{1:T}}|\mathbf{x_0}) = \prod_{t=1}^T q(\mathbf{x_t}|\mathbf{x_{t-1}})$ denotes the transition probability at each step. The reverse process is modeled as $p_\theta(\mathbf{x_{0:T}}|g) = p(\mathbf{x_T}) \prod_{t=1}^T p_\theta(\mathbf{x_{t-1}}|\mathbf{x_t}, g)$, with θ representing the model parameters. The training objective is to match $p_\theta(\mathbf{x_0}|g)$ with the data distribution $q(\mathbf{x_0}|g)$, optimized by minimizing the variational upper bound of the negative log-likelihood:

$$L(\theta) = \mathbb{E}_q\Big[-\log p_\theta(\mathbf{x_0}|\mathbf{x_1}, g) + \sum_{t=2}^T D_{KL}(q(\mathbf{x_{t-1}}|\mathbf{x_t}, \mathbf{x_0}) \| p_\theta(\mathbf{x_{t-1}}|\mathbf{x_t}, g))\Big] \tag{5}$$

More details are described in Appendix.

2.4. Decoder for feasiblity in ML-based CO solvers

As we mentioned in Section 2.1, the feasible solution space $\mathcal{F}_g$ is a much smaller subset compared to the total solution space $\mathcal{X}_g$. However, the above diffusion model only guarantees feasibility of the sampled solution $\mathbf{x_0}$. In other words, we know that $\mathbf{x_0}$ belongs to the total solution space $\mathcal{X}_g$ but may not belong to the feasible solution space $\mathcal{F}_g$. To overcome this issue, Sun & Yang (2023), who suggest the diffusion models for CO, introduce an additional post-processing decoder $f_g : \mathcal{X}_g \to \mathcal{F}_g$ which slightly changes the sampled solution $\mathbf{x}_0$ into the feasible solution $f_g(\mathbf{x}_0)$ near the original solution, i.e., $\mathbf{x}_0 \approx f_g(\mathbf{x}_0)$. Since the decoder f_g modifies the solution, the goal in CO is also changed from minimizing cost$(\mathbf{x}_0, g) +$ valid$(\mathbf{x}_0, g)$ to minimizing cost$(f_g(\mathbf{x}_0), g)$. However, the previous work (Sun & Yang, 2023) introduces a diffusion model for CO but does not consider the effect of the decoder in their learning objective in (5).

In RL, it is much easier to consider these decoding mechanisms without hurting the objective compared to the CO scenario. In the RL scenario, we can slightly modify the objective $R(\theta)$ in (4) as follows:

$$R(\theta) = \mathbb{E}_{g\sim \mathbf{P}(g),\, \mathbf{x}\sim p_\theta(\mathbf{x}|g)}[-\text{cost}(f_g(\mathbf{x}_0), g)].$$

With this simple modification of $R(\theta)$, the solver knows how the objective is affected by the decoder f_g since the cost is determined by the post-processed solution $f_g(\mathbf{x}_0, g)$. Taking these factors into account, we introduce an MDP in the following section which formulates the denoising process of the pretrained diffusion model with the consideration of the decoder f_g.

3. Method

3.1. MDP modeling of diffusion model for CO

An MDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, P, \rho_0, R)$, where $\boldsymbol{s} \in \mathcal{S}$ is a state in the state space $\mathcal{S}$, $\mathbf{a} \in \mathcal{A}$ is an action belongs

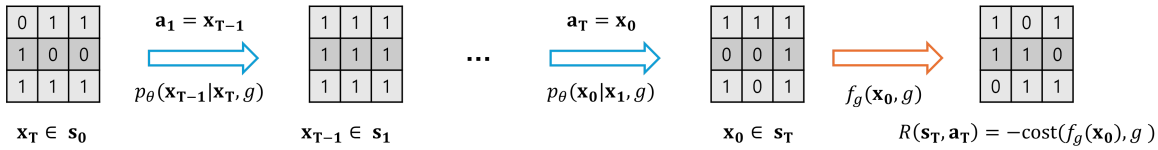

Figure 2. The overall denoise process in terms of MDP. The initial random noise $\mathbf{x}_T$ is sampled from the Bern$(\boldsymbol{p} = 0.5^N)$.

to the action space $\mathcal{A}$, $P(\mathbf{s}_{t+1}|\, \mathbf{s}_t, \mathbf{a}_t)$ is the state transition distribution, $\rho_0(\mathbf{s}_0)$ is the initial state distribution, and $R(\mathbf{s}_t, \mathbf{a}_t)$ is the reward function. The objective of RL is to learn a policy π that maximizes the expected cumulative reward $J(\pi)$, formalized as $\mathbb{E}_{\tau \sim p(\tau|\pi)} \left[\sum_{t=0}^{T} R\left(\mathbf{s}_t, \mathbf{a}_t\right)\right]$ where $\tau = (\mathbf{s}_0, \mathbf{a}_0 ... \mathbf{s}_T, \mathbf{a}_T)$ is a sequence of states and actions from a policy in the MDP.

We formulate the denoising process in the diffusion process as Markov Decision Process (MDP) for CO, motivated from (Black et al., 2024) in the image domain:

$$\begin{aligned}
\mathbf{s}_t &\triangleq \left(g, t, \mathbf{x}_t\right), \\
\mathbf{a}_t &\triangleq \mathbf{x}_{t-1}, \\
\pi\left(\mathbf{a}_t \mid \mathbf{s}_t\right) &\triangleq p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, g\right), \\
P\left(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t\right) &\triangleq \left(\delta_{\mathbf{s}}, \delta_{t-1}, \delta_{\mathbf{x}_{t-1}}\right), \\
\rho_0(\mathbf{s}_0) &\triangleq (g, t, \text{Bern}(\boldsymbol{p} = 0.5^{N_g})), \\
R\left(\mathbf{s}_t, \mathbf{a}_t\right) &\triangleq \begin{cases} -c_s\left(f_g(\mathbf{x}_0), g\right) & \text{if } t = 0, \\ 0 & \text{otherwise.} \end{cases}
\end{aligned} \tag{6}$$

where Bern$(\boldsymbol{p})$ is a Bernoulli distribution with vector probabilities $\mathbf{p}$ that samples the initial random noise $\mathbf{x}_T$, and δ_y is the Dirac delta distribution with nonzero density only at y. We then apply a policy gradient algorithm for optimizing the iterative denoising procedure with the cost function:

$$\nabla_\theta J = \mathbb{E}\left[\sum_{t=0}^{T} \nabla_\theta \log p_\theta\left(\mathbf{x}_{t-1} \mid \mathbf{x}_t, g\right)\left(-\text{cost}\left(f_g(\mathbf{x}_0), g\right)\right)\right] \tag{7}$$

We note that the solver is able to consider the effect of the post-processed solution $\boldsymbol{x}_0$ by the decoder f_g if the agent learns the suggested MDP properly.

Remark 3.1. One of the challenges in current RL-finetuning techniques is reward hacking (Black et al., 2024; Fan et al., 2023), where the model learns to optimize for high reward values while sacrificing actual output quality, which is hard to compute. However, in our problem domain of combinatorial optimization (CO), the decoder inherently guarantees feasibility. This means that minimizing the cost function directly correlates with improving the solution quality. As a result, our approach is not susceptible to the reward hacking issue that plagues many existing RL-finetuning methods.

3.2. RL fine-tuning for cost-aware diffusion model

The overall structure of our framework is illustrated in Figure 2. CADO consists of two phases. In the first phase, the diffusion model is trained using the given dataset with the supervised learning objective objective $L(\theta)$ in (5). In the second phase, we apply RL fine-tuning on the pretrained diffusion model to optimize $R(\theta)$ in (4). During this phase, the training instances g can be newly generated from the distribution $\boldsymbol{P}(g)$ or sampling from the instances in the train dataset. In general, generating unseen instances is more beneficial for aspects such as generalization compared to using existing instances in the train dataset. Therefore, when the distribution of instances is known, we sampled new instances directly from the environment rather than utilizing the existing instances.

To accurately measure the effectiveness of RL-finetuning compared to previous works (Sun & Yang, 2023; Li et al., 2023) with the diffusion models that did not consider Rl-finetuing, we directly finetune the pretrained diffusion model employed in those papers and performed finetuning. The denoising component of the Diffusion model consists of a 12-layer anisotropic Graph Neural Network (GNN), with each layer utilizing either 128 or 256 units. More details of the network architecture are described in Appendix.

Many other techniques in (Sun & Yang, 2023; Li et al., 2023) are also applied in a similar manner. The decoders used for post-processing to generate feasible solutions are identical to those employed in (Sun & Yang, 2023; Li et al., 2023). The 2OPT heuristic(Lin & Kernighan, 1973) can be optionally applied for TSP. Finally, we adopted the local rewriting technique from Chen & Tian (2019a); Li et al. (2023), which involves reinjecting noise and performing denoising after the initial denoising process.

Recently, Li et al. (2023) proposed a method called T2T that enables DIFUSCO to utilize cost information by guiding the denoising steps through gradients from the differentiable cost function, thereby directly minimizing the CO objective during inference. (Li et al., 2023) is similar to our work in that it also incorporates cost information. However, our approach does not require a differentiable cost function, which is not always straightforward to define in CO problems.

During RL fine-tuning, we apply several techniques to facilitate efficient training. First, we freeze the first 11 layers

Table 1. Results on TSP-50 and TSP-100. AS: Active Search, S: Sampling Decoding, BS: Beam Search, RRC: Random Re-Construct(algorithm from Luo et al. (2023), which iteratively refines the partial solution). * represents the baseline for computing the drop. All the results except for † and T2T are taken from Li et al. (2023). The results of models† are taken from Zhou et al. (2024), which are evaluated on the different test instance set with others.

Algorithm	Type	TSP-50		TSP-100	
		Length ↓	Drop ↓	Length ↓	Drop ↓
Concorde (Applegate et al., 2006)	Exact	5.69*	0.00%	7.76*	0.00%
2OPT (Lin & Kernighan, 1973)	Heuristics	5.86	2.95%	8.03	3.54%
Farthest Insertion	Heuristics	6.12	7.50%	8.72	12.36%
AM (Kool et al., 2019b)	RL+Grdy	5.80	1.76%	8.12	4.53%
GCN (Joshi et al., 2019a)	SL+Grdy	5.87	3.10%	8.41	8.38%
Transformer (Bresson & Laurent, 2021)	RL+Grdy	5.71	0.31%	7.88	1.42%
POMO (Kwon et al., 2020)	RL+Grdy	5.73	0.64%	7.84	1.07%
Sym-NCO (Kim et al., 2022)	RL+Grdy	-	-	7.84	0.94%
Image Diffusion (Graikos et al., 2022b)	SL+Grdy	5.76	1.23%	7.92	2.11%
BQ† (Drakulic et al., 2023)	SL+Grdy	-	-	7.79	0.35%
LEHD† (Luo et al., 2023)	SL+Grdy	-	-	7.81	0.58%
ICAM† (Zhou et al., 2024)	RL+Grdy	-	-	7.83	0.90%
DIFUSCO (Sun & Yang, 2023)	SL+Grdy	5.72	0.48%	7.84	1.01%
T2T (Sun & Yang, 2023)	SL+Grdy	5.69	0.04%	7.77	0.18%
CADO (Ours)	SL+RL+Grdy	**5.69**	**0.01%**	**7.77**	**0.08%**
AM (Kool et al., 2019b)	RL+Grdy+2OPT	5.77	1.41%	8.02	3.32%
GCN (Joshi et al., 2019a)	SL+Grdy+2OPT	5.70	0.12%	7.81	0.62%
Transformer (Bresson & Laurent, 2021)	RL+Grdy+2OPT	5.70	0.16%	7.85	1.19%
POMO (Kwon et al., 2020)	RL+Grdy+2OPT	5.73	0.63%	7.82	0.82%
Sym-NCO (Kim et al., 2022)	RL+Grdy+2OPT	-	-	7.82	0.76%
BQ† (Drakulic et al., 2023)	-	-	-	-	-
LEHD† (Luo et al., 2023)	SL+Grdy+RRC	-	-	7.76	0.01%
ICAM† (Zhou et al., 2024)	RL+Grdy+RRC	-	-	7.79	0.41%
DIFUSCO (Sun & Yang, 2023)	SL+Grdy+2OPT	5.69	0.09%	7.78	0.22%
T2T (Li et al., 2023)	SL+Grdy+2OPT	5.69	0.02%	7.76	0.06%
CADO (Ours)	SL+RL+Grdy+2OPT	**5.69**	**0.00%**	**7.76**	**0.01%**

in the GNN architecture and only update the parameters in the last layer of the GNN. Additionally, we optionally apply Low-Rank Adaptation (LoRA) (Hu et al., 2022) to fine-tune the remaining 11 layers. Our experimental results demonstrate that for most tasks, fine-tuning only the last layer is sufficient, leading to faster training speed and reduced memory usage. However, in certain cases, applying LoRA yielded significantly improved performance. We observe that LoRA was particularly beneficial when the performance of the pre-trained diffusion model alone was inadequate. In such scenarios, incorporating LoRA contributed to notable performance enhancements.

4. Experiment

The experiments were carried out using a single NVIDIA Telsla A40 GPU and two cpu cores of AMD EPYC 7413 24-Core Processor both for training and testing. Basically, all test procedures are the same as DIFUSCO (Sun & Yang, 2023) and T2T (Li et al., 2023) studies, which serve as crucial baselines for comparison.

4.1. Experiment settings

Problems We test our proposed CADO on the Traveling Salesman Problem (TSP) and the Maximal Independent Set (MIS), which are basically edge and node selecting problem respectively. TSP is the most commonly used benchmark combinatorial optimization problem, where the objective is to determine the shortest possible route that visits a set of nodes exactly once and returns to the original node. $\text{cost}(\boldsymbol{x}, G)$ in the Equation 1 is defined as $\text{cost}_{\text{TSP}}(\mathbf{x}, G) = \sum_{i,j} \mathbf{x}_{i,j} \cdot w_{i,j}$, where $w_{i,j}$ denotes the weight (distance) between vertices i and j, and $\text{valid}_{\text{TSP}}(\mathbf{x}, G)$ returns 0 only when $\mathbf{x}$ visits a set of nodes exactly once and returns to the original node.

MIS is another widely used benchmark problem where the objective is to find the largest subset of vertices in a graph such that no two vertices in the subset are adjacent. The cost is defined as $\text{cost}_{\text{MIS}}(\mathbf{x}, G) = \sum_i (1 - \mathbf{x}_i)$, and $\text{valid}_{\text{MIS}}(\mathbf{x}, G)$ returns 0 only when each vertice in the set has no connection to any other vertice in the set.

Table 2. Results on TSP-500 and TSP-1000. AS: Active Search, S: Sampling Decoding, BS: Beam Search, RRC: Random Re-Construct(algorithm from Luo et al. (2023), which iteratively refines the partial solution). * represents the baseline for computing the drop. All the results except for † and T2T are taken from Li et al. (2023). The results of models† are taken from Zhou et al. (2024), which are evaluated on the different test instance set with others.

Algorithm	Type	TSP-500			TSP-1000		
		Length ↓	Drop ↓	Time	Length ↓	Drop ↓	Time
Concorde (Applegate et al., 2006)	Exact	16.55*	-	37.66m	23.12*	-	6.65h
Gurobi (Gurobi Optimization, 2020)	Exact	16.55	0.00%	45.63h	-	-	-
LKH-3 (default) (Helsgaun, 2017)	Heuristics	16.55	0.00%	46.28m	23.12	0.00%	2.57h
Farthest Insertion	Heuristics	18.30	10.57%	0s	25.72	11.25%	0s
AM (Kool et al., 2019b)	RL+Grdy	20.02	20.99%	1.51m	31.15	34.75%	3.18m
GCN (Joshi et al., 2019a)	SL+Grdy	29.72	79.61%	6.67m	48.62	110.29%	28.52m
POMO+EAS-Emb (Hottung et al., 2021)	RL+AS+Grdy	19.24	16.25%	12.80h	-	-	-
POMO+EAS-Tab (Hottung et al., 2021)	RL+AS+Grdy	24.54	48.22%	11.61h	49.56	114.36%	63.45h
DIMES (Qiu et al., 2022)	RL+Grdy	18.93	14.38%	0.97m	26.58	14.97%	2.08m
DIMES (Qiu et al., 2022)	RL+AS+Grdy	17.81	7.61%	2.10h	24.91	7.74%	4.49h
DIMES (Qiu et al., 2022)	RL+Grdy+2OPT	17.65	6.62%	1.01m	24.83	7.38%	2.29m
DIMES (Qiu et al., 2022)	RL+AS+Grdy+2OPT	17.31	4.57%	2.10h	24.33	5.22%	4.49h
BQ† (Drakulic et al., 2023)	SL+Grdy	16.72	1.18%	0.77m	23.65	2.27%	1.9m
LEHD† (Luo et al., 2023)	SL+Grdy	16.78	1.56%	0.27m	23.85	3.17%	1.6m
LEHD† (Luo et al., 2023)	SL+Grdy+RRC	16.58	0.34%	8.7m	23.40	1.20%	48.6m
ICAM† (Zhou et al., 2024)	RL+Grdy	16.78	1.56%	0.03	23.80	2.93%	0.03m
ICAM† (Zhou et al., 2024)	RL+Grdy+RRC	16.69	1.01%	2.4m	23.55	1.86%	16.8m
DIFUSCO (Sun & Yang, 2023)	SL+Grdy	18.11	9.41%	5.70m	25.72	11.24%	17.33m
DIFUSCO (Sun & Yang, 2023)	SL+Grdy+2OPT	16.81	1.55%	5.75m	23.55	1.86%	17.52m
T2T (Li et al., 2023)	SL+Grdy	17.69	6.92%	4.90m	25.39	9.83%	17.93m
T2T (Li et al., 2023)	SL+G+2OPT	16.68	0.83%	4.83m	23.41	1.26%	18.37m
CADO (Ours)	SL+RL+Grdy	16.97	2.56%	2.52m	24.92	7.78 %	18.31m
CADO (Ours)	SL+RL+Grdy+2OPT	**16.64**	**0.58**%	2.67m	**23.35**	**1.02** %	7.67m
EAN (Deudon et al., 2018)	RL+S+2OPT	23.75	43.57%	57.76m	47.73	106.46%	5.39h
AM (Kool et al., 2019b)	RL+BS	19.53	18.03%	21.99m	29.90	29.23%	1.64h
GCN (Joshi et al., 2019a)	SL+BS	30.37	83.55%	38.02m	51.26	121.73%	51.67m
DIMES (Qiu et al., 2022)	RL+S	18.84	13.84%	1.06m	26.36	14.01%	2.38m
DIMES (Qiu et al., 2022)	RL+AS+S	17.80	7.55%	2.11h	24.89	7.70%	4.53h
DIMES (Qiu et al., 2022)	RL+S+2OPT	17.64	6.56%	1.10m	24.81	7.29%	2.86m
DIMES (Qiu et al., 2022)	RL+AS+S+2OPT	17.29	4.48%	2.11h	24.32	5.17%	4.53h
BQ† (Drakulic et al., 2023)	SL+BS	16.62	0.58%	11.9m	23.43	1.36%	29.4m
ICAM† (Zhou et al., 2024)	RL+BS	16.69	1.01%	1.5m	23.54	1.83%	10.5m
ICAM† (Zhou et al., 2024)	RL+S	16.65	0.78%	0.63m	23.49	1.58%	3.8m
DIFUSCO (Sun & Yang, 2023)	SL+S	17.48	5.65%	19.02m	25.11	8.61%	59.18m
DIFUSCO (Sun & Yang, 2023)	SL+S +2OPT	16.69	0.37%	19.05m	23.42	1.30%	59.53m
T2T (Li et al., 2023)	SL+S	17.14	3.60%	17.05m	24.85	7.51%	1.12h
T2T (Li et al., 2023)	SL+S +2OPT	16.62	0.46%	17.02m	23.31	0.85%	1.17h
CADO (Ours)	SL+RL+S	16.75	1.27%	6.83m	24.47	5.88 %	24.73m
CADO (Ours)	SL+RL+S+2OPT	**16.60**	0.34%	6.90m	**23.28**	0.69 %	25.78m

Table 3. Comparison of Different Algorithms on SATLIB and ER-[700-800]

Algorithm	Type	SATLIB			ER-[700-800]		
		Size ↑	Drop ↓	Time	Size ↑	Drop ↓	Time
KaMIS (Lamm et al., 2016)	Heuristics	425.96*	-	37.58m	44.87*	-	52.13m
Gurobi (Gurobi Optimization, 2020)	Exact	425.95	0.00%	26.00m	41.28	7.78%	50.00m
Intel (Li et al., 2018a)	SL+Grdy	420.66	1.48%	23.05m	34.86	22.31%	6.06m
DIMES (Qiu et al., 2022)	RL+Grdy	421.24	1.11%	24.17m	38.24	14.78%	6.12m
DIFUSCO (Sun & Yang, 2023)	SL+Grdy	424.56	0.33%	8.25m	36.55	18.53%	8.82m
T2T (Li et al., 2023)	SL+Grdy	**425.02**	**0.22%**	8.12m	39.56	11.83%	8.53m
CADO (Ours)	SL+RL+Grdy	425.01	0.22%	9.52m	**42.96**	**4.25%**	9.50m
Intel (Li et al., 2018a)	SL+TS	-	-	-	38.80	13.43%	20.00m
DGL (Böther et al., 2022)	SL+TS	-	-	-	37.26	16.96%	22.71m
LwD (Ahn et al., 2020a)	RL+S	422.22	0.88%	18.83m	41.17	8.25%	6.33m
GFlowNets (Zhang et al., 2023)	UL+S	423.54	0.57%	23.22m	41.14	8.53%	2.92m
DIFUSCO (Sun & Yang, 2023)	SL+S	425.13	0.19%	26.32m	40.35	10.07%	32.98m
T2T (Li et al., 2023)	SL+S	**425.22**	**0.17%**	23.80m	41.37	7.81%	29.73m
CADO (Ours)	SL+RL+S	425.14	0.19%	16.57m	**43.53**	**2.998%**	11.90m

Datasets In TSP experiments, we use the training instances provided by DIFUSCO (Sun & Yang, 2023) where the solutions are generated by the Concorde exact solver (Applegate et al., 2006) or the LKH-3 heuristic solver (Helsgaun, 2017). For the fair comparison, we use the same test instances as in Joshi et al. (2022); Kool et al. (2019b) for TSP-50/100 and Fu et al. (2021b) for TSP-500/1000. In MIS experiments, we experiment on two types of graphs following (Li et al., 2018b; Ahn et al., 2020b; Böther et al., 2022; Qiu et al., 2022; Sun & Yang, 2023; Li et al., 2023), SATLIB (Hoos & Stutzle, 2000) and Erdős–Rényi (Erdos & Renyi, 1960). We also use the training instances provided by DIFUSCO, and test instances from Qiu et al. (2022).

Evaluation Metrics We evaluate our model and other baselines in terms of three metrics : 1) Length : the average tour length for TSP (the smaller, the better), and Size : the average size of independent set for MIS (the larger, the better). 2) Drop : the average performance difference between the generated solutions from the models and optimal solutions. 3) Time : the total run time during test time.

Baselines We compare our method with the following methods : (1) Classical Solvers: Concorde [2], LKH3 [16], HGS [50], and OR-Tools [41]; (2) Constructive NCO: POMO [29], MDAM [54], EAS [19], SGBS [8], and BQ [12]; (3) Heatmap-based Method: Att-GCN+MCTS [13].

4.2. Main Result

TSP 50/100 Our experiments on TSP 50 and TSP 100, summarized in Table 1. By utilizing reward signals during training, we significantly improve the model's performance, achieving the state-of-the-art (SOTA). Notably, for TSP 50, our model without 2-opt heuristics (SL+RL+Grdy, drop: 0.01%) outperforms DIFUSCO (0.09%) and T2T (0.02%) with 2-opt, underscoring the superior optimization capability of our RL fine-tuning approach.

TSP 500/1000 For larger instances, our model continues to deliver impressive results. The message remains consistent: our fine-tuning approach significantly reduces the gap, emphasizing its effectiveness. Our method consistently achieves SOTA performance, validating the effectiveness of combining supervised and RL losses during training. The success of our method can be attributed to its ability to observe multiple new instances due to RL fine-tuning, and incorporating the post-processing decoder in the training phase, allowing the model to learn to produce solutions that are optimal for the post-processing decoder. Note that the computational costs with ours and T2T are the same and very similar to DIFUSCO, but different library versions and optimized code result in different computation times.

MIS SAT/ER Our experimental results on the SATLIB dataset shows competitive performance with previous state-of-the-art, T2T. As the performance of DIFUSCO, our backbone model, is already near optimal, only the minimal improvement has been achieved. Additionally, generating new SAT instances is not trivial. As a result, we utilized the existing training dataset during RL fine-tuning, which might have limited potential performance improvements. This finding suggests that for better results, exposing the RL fine-tuning process to new samples, rather than reusing the samples employed in supervised learning (SL), could lead to more significant performance enhancements. The performance on the ER dataset is outstanding. Our CADO approach achieved a maximum independent set size of 42.96 with a drop of 4.25%, significantly better than the results of the previous state-of-the-art. In this case, we generated random graphs during RL fine-tuning, which contributed to dramatic improvements in performance.

4.3. SL under the low quality train dataset

Table 4. Results on the low quality dataset.

Algorithm	**Drop 0%** Drop ↓	**Drop 1.36%** Drop ↓
DIFUSCO	0.48 %	2.298%
T2T	0.04%	1.001%
CADO(ours)	0.01 %	0.911%

A significant advantage of RL fine-tuning is its ability to continually explore higher quality solutions during training. Therefore, it can be less sensitive to the quality of the given dataset. To verify this, we constructed an additional dataset consisting only of suboptimal solutions for TSP100, in addition to the dataset with optimal solutions. The suboptimal dataset was created by running LKH-3 for 1 second per instance, resulting in samples with an average drop of 1.36% compared to the optimal dataset. Table 4 shows the performance of algorithms trained on both the optimal dataset (Drop 0%) and the suboptimal dataset (Drop 1.36%). As expected, DIFUSCO's performance significantly decreased when trained on the lower-quality dataset. In contrast, both our approach and T2T, which utilize cost information, demonstrated the ability to generate samples of higher quality than the provided dataset. Our method slightly outperformed T2T. These results highlight the importance of leveraging cost information in combinatorial optimization.

4.4. Transfer learning

Table 5. Results on transfer learning on various TSP size.

Fine-tuning	**100→500** Drop ↓	**500→1000** Drop ↓
SL → ×	3.2%	2.12%
SL → SL	1.55%	1.86%
SL → RL	1.59%	1.04%

In this section, we conducted experiments in a transfer learning setting where the tasks of the training data and the target task differ. While it is possible to fine-tune using RL as we have done, if a dataset for the target task exists, it is also feasible to fine-tune using SL as DIFUSCO does. We compared SL fine-tuning and RL fine-tuning in this context. We set up two environments: one where the model was trained on TSP100 and then fine-tuned on TSP500 (100→500), and another where the model was trained on TSP500 and then fine-tuned on TSP1000 (500→1000). As shown in the Table 5, directly applying the model without fine-tuning results in poor performance. Compared to SL fine-tuning, our method achieved similar performance on TSP500 and better performance on TSP1000, despite not using an additional dataset labeled with solutions close to optimal. These results demonstrate that RL fine-tuning is more cost-effective and efficient.

5. Conclusion

In this paper, we introduced an RL fine-tuning framework for generative models in Combinatorial Optimization (CO) problems, addressing the limitations of traditional diffusion-based solvers. Our approach integrates cost-awareness into solution generation, significantly enhancing performance in various CO domains. Furthermore, it shows robustness with the quality of the training data, and can effectively adapts to different scales of CO problems through transfer learning. These overall results suggest that our integration of cost information and the decoding process into the learning framework offers a promising improvement for generative model-based CO solvers.

References

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, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 134–144. PMLR, 2020a. URL http://proceedings.mlr.press/v119/ahn20a.html.

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, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 134–144. PMLR, 2020b.

Applegate, D., Bixby, R., Chvatal, V., and Cook, W. Concorde tsp solver. http://www.math.uwaterloo.ca/tsp/concorde.html, 2006.

Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and Van Den Berg, R. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021a.

Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and van den Berg, R. Structured denoising diffusion models in discrete state-spaces. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 17981–17993. Curran Associates, Inc., 2021b.

Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., Joseph, N., Kadavath, S., Kernion, J., Conerly, T., Showk, S. E., Elhage, N., Hatfield-Dodds, Z., Hernandez, D., Hume, T., Johnston, S., Kravec, S., Lovitt, L., Nanda, N., Olsson, C., Amodei, D., Brown, T. B., Clark, J., McCandlish, S., Olah, C., Mann, B., and Kaplan, J. Training a helpful and harmless assistant with reinforcement learning from human feedback. *CoRR*, abs/2204.05862, 2022. doi: 10.48550/ARXIV.2204.05862.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., and Bengio, S. Neural combinatorial optimization with reinforcement learning. *CoRR*, abs/1611.09940, 2016. URL http://arxiv.org/abs/1611.09940.

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

Black, K., Janner, M., Du, Y., Kostrikov, I., and Levine, S. Training diffusion models with reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

Böther, M., Kißig, O., Taraz, M., Cohen, S., Seidel, K., and Friedrich, T. What's wrong with deep learning in tree search for combinatorial optimization. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Bresson, X. and Laurent, T. An experimental study of neural networks for variable graphs. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net, 2018a.

Bresson, X. and Laurent, T. An experimental study of neural networks for variable graphs. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net, 2018b.

Bresson, X. and Laurent, T. The transformer network for the traveling salesman problem. *CoRR*, abs/2103.03012, 2021.

Chen, X. and Tian, Y. Learning to perform local rewriting for combinatorial optimization. *Advances in neural information processing systems*, 32, 2019a.

Chen, X. and Tian, Y. Learning to perform local rewriting for combinatorial optimization. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019b.

Clark, K., Vicol, P., Swersky, K., and Fleet, D. J. Directly fine-tuning diffusion models on differentiable rewards. In *The Twelfth International Conference on Learning Representations*, 2024.

da Costa, P. R. d. O., Rhuggenaath, J., Zhang, Y., and Akcay, A. Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning. In Pan, S. J. and Sugiyama, M. (eds.), *Proceedings of The 12th Asian Conference on Machine Learning*, volume 129 of *Proceedings of Machine Learning Research*, pp. 465–480. PMLR, 18–20 Nov 2020.

Deng, M., Wang, J., Hsieh, C., Wang, Y., Guo, H., Shu, T., Song, M., Xing, E. P., and Hu, Z. Rlprompt: Optimizing discrete text prompts with reinforcement learning. In Goldberg, Y., Kozareva, Z., and Zhang, Y. (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 3369–3391. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.EMNLP-MAIN.222. URL https://doi.org/10.18653/v1/2022.emnlp-main.222.

Deudon, M., Cournut, P., Lacoste, A., Adulyasak, Y., and Rousseau, L.-M. Learning heuristics for the tsp by policy gradient. In *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pp. 170–181. Springer, 2018.

Drakulic, D., Michel, S., Mai, F., Sors, A., and Andreoli, J. BQ-NCO: bisimulation quotienting for efficient neural combinatorial optimization. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Erdos, P. and Renyi, A. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci.*, 7:17, 1960.

Fan, Y., Watkins, O., Du, Y., Liu, H., Ryu, M., Boutilier, C., Abbeel, P., Ghavamzadeh, M., Lee, K., and Lee, K. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 79858–79885. Curran Associates, Inc., 2023.

Fu, Z., Qiu, K., and Zha, H. Generalize a small pre-trained model to arbitrarily large TSP instances. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 7474–7482. AAAI Press, 2021a. doi: 10.1609/AAAI.V35I8.16916. URL https://doi.org/10.1609/aaai.v35i8.16916.

Fu, Z., Qiu, K., and Zha, H. Generalize a small pre-trained model to arbitrarily large TSP instances. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 7474–7482. AAAI Press, 2021b. doi: 10.1609/AAAI.V35I8.16916.

Geisler, S., Sommer, J., Schuchardt, J., Bojchevski, A., and Günnemann, S. Generalization of neural combinatorial solvers through the lens of adversarial robustness. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=vJZ7dPIjip3.

Graikos, A., Malkin, N., Jojic, N., and Samaras, D. Diffusion models as plug-and-play priors. In *NeurIPS*, 2022a.

Graikos, A., Malkin, N., Jojic, N., and Samaras, D. Diffusion models as plug-and-play priors. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022b.

Gurobi Optimization, L. Gurobi optimizer reference manual. http://www.gurobi.com, 2020.

Helsgaun, K. An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University*, pp. 24–50, 2017.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Hoogeboom, E., Nielsen, D., Jaini, P., Forré, P., and Welling, M. Argmax flows and multinomial diffusion: Learning categorical distributions. *Advances in Neural Information Processing Systems*, 34:12454–12465, 2021.

Hoos, H. and Stutzle, T. Satlib: An online resource for research on sat. In van Maaren I. P. Gent, H. and Walsh, T. (eds.), *SAT2000*, pp. 283–292. IOS Press, 2000.

Hottung, A., Kwon, Y.-D., and Tierney, K. Efficient active search for combinatorial optimization problems. *arXiv preprint arXiv:2106.05126*, 2021.

Hou, Q., Yang, J., Su, Y., Wang, X., and Deng, Y. Generalize learned heuristics to solve large-scale vehicle routing problems in real-time. In *International Conference on Learning Representations*, 2023.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Bach, F. R. and Blei, D. M. (eds.), *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pp. 448–456. JMLR.org, 2015.

Joshi, C. K., Laurent, T., and Bresson, X. An efficient graph convolutional network technique for the travelling salesman problem. *arXiv preprint arXiv:1906.01227*, 2019a.

Joshi, C. K., Laurent, T., and Bresson, X. An efficient graph convolutional network technique for the travelling salesman problem. *CoRR*, abs/1906.01227, 2019b. URL http://arxiv.org/abs/1906.01227.

Joshi, C. K., Cappart, Q., Rousseau, L., and Laurent, T. Learning the travelling salesperson problem requires rethinking generalization. *Constraints An Int. J.*, 27(1-2): 70–98, 2022. doi: 10.1007/S10601-022-09327-Y.

Karp, R. M. On the computational complexity of combinatorial problems. *Networks*, 5(1):45–68, 1975.

Kim, M., Park, J., and Park, J. Sym-nco: Leveraging symmetricity for neural combinatorial optimization. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, 2022.

Kool, W., Hoof, H., and Welling, M. Learning a latent search space for routing problems using variational autoencoders. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019a.

Kool, W., Van Hoof, H., and Welling, M. Attention, learn to solve routing problems! In *International Conference on Learning Representations (ICLR)*, 2019b.

Krizhevsky, A. Convolutional deep belief networks on cifar-10. 2010.

Kwon, Y.-D., Choo, J., Kim, B., Yoon, I., Min, S., and Gwon, Y. Pomo: Policy optimization with multiple optima for reinforcement learning. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 2020.

Lamm, S., Sanders, P., Schulz, C., Strash, D., and Werneck, R. F. Finding near-optimal independent sets at scale. In Goodrich, M. T. and Mitzenmacher, M. (eds.), *Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments, ALENEX 2016, Arlington, Virginia, USA, January 10, 2016*, pp. 138–150. SIAM, 2016. doi: 10.1137/1.9781611974317.12.

Li, S., Yan, Z., and Wu, C. Learning to delegate for large-scale vehicle routing. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 26198–26211. Curran Associates, Inc., 2021.

Li, Y., Guo, J., Wang, R., and Yan, J. From distribution learning in training to gradient search in testing for combinatorial optimization. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Li, Z., Chen, Q., and Koltun, V. Combinatorial optimization with graph convolutional networks and guided tree search. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 537–546, 2018a.

Li, Z., Chen, Q., and Koltun, V. Combinatorial optimization with graph convolutional networks and guided tree search. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 537–546, 2018b.

Lin, S. and Kernighan, B. W. An effective heuristic algorithm for the traveling-salesman problem. *Oper. Res.*, 21 (2):498–516, 1973. doi: 10.1287/OPRE.21.2.498.

Luo, F., Lin, X., Liu, F., Zhang, Q., and Wang, Z. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Mirhoseini, A., Pham, H., Le, Q. V., and Bengio, S. The policy-gradient placement and generative routing neural networks for chip design. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Niu, C., Song, Y., Song, J., Zhao, S., Grover, A., and Ermon, S. Permutation invariant graph generation via score-based generative modeling. In Chiappa, S. and Calandra, R. (eds.), *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pp. 4474–4484. PMLR, 2020.

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

Qiu, R., Sun, Z., and Yang, Y. Dimes: A differentiable meta solver for combinatorial optimization problems. *arXiv preprint arXiv:2210.04123*, 2022.

Sun, Z. and Yang, Y. Difusco: Graph-based diffusion solvers for combinatorial optimization. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine,

S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 3706–3731. Curran Associates, Inc., 2023.

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

Wu, Y., Song, W., Cao, Z., Zhang, J., and Lim, A. Learning improvement heuristics for solving routing problems. *IEEE Transactions on Neural Networks and Learning Systems*, 33:5057–5069, 2019.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Zhang, D., Dai, H., Malkin, N., Courville, A. C., Bengio, Y., and Pan, L. Let the flows tell: Solving graph combinatorial optimization problems with gflownets. *CoRR*, abs/2305.17010, 2023. doi: 10.48550/ARXIV.2305.17010. URL https://doi.org/10.48550/arXiv.2305.17010.

Zhou, C., Lin, X., Wang, Z., Tong, X., Yuan, M., and Zhang, Q. Instance-conditioned adaptation for large-scale generalization of neural combinatorial optimization, 2024.

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P. F., and Irving, G. Fine-tuning language models from human preferences. *CoRR*, abs/1909.08593, 2019. URL http://arxiv.org/abs/1909.08593.

A. Diffusion Loss

In CO, considering that the entry of the optimization variable $\boldsymbol{x}$ are indicators of whether to select a node or an edge, each entry can also be represented as an one-hot $\{0,1\}^2$ while modeling it with Bernoulli distribution. Therefore, for diffusion process, $\boldsymbol{x}$ turns into N one-hot vectors $\mathbf{x_0} \in \{0,1\}^{N\times 2}$. Then, discrete diffusion model (Austin et al., 2021b) is utilized. Specifically, at each time step t, the process transitions from $\mathbf{x_{t-1}}$ to $\mathbf{x_t}$ defined as:

$$q(\mathbf{x_t}|\mathbf{x_{t-1}}) = \mathrm{Cat}(\mathbf{x_t}; \mathbf{p} = \boldsymbol{x}_{t-1}\mathbf{Q_t}) \tag{8}$$

where the $\mathrm{Cat}(\boldsymbol{x}; \mathbf{p})$ is a categorical distribution over $x \in \{0,1\}^{N\times 2}$ with vector probabilities $\mathbf{p}$ and transition probability matrix $\mathbf{Q_t}$ is:

$$\mathbf{Q_t} = \begin{bmatrix} (1-\beta_t) & \beta_t \\ \beta_t & (1-\beta_t) \end{bmatrix} \tag{9}$$

Here, β_t represents the noise level at time t. The t-step marginal distribution can be expressed as:

$$q(\mathbf{x_t}|\mathbf{x_0}) = \mathrm{Cat}(\mathbf{x_t}; \mathbf{p} = \mathbf{x_0}\overline{\mathbf{Q}}_\mathbf{t}) \tag{10}$$

where $\overline{\mathbf{Q}}_\mathbf{t} = \mathbf{Q_1}\mathbf{Q_2}, \ldots, \mathbf{Q}_t$. To obtain the distribution $q(\mathbf{x_{t-1}}|\mathbf{x_t}, \mathbf{x_0})$ for the reverse process, Bayes' theorem is applied, resulting in:

$$q(\mathbf{x_{t-1}}|\mathbf{x_t}, \mathbf{x_0}) = \mathrm{Cat}\left(\mathbf{x_{t-1}}; \mathbf{p} = \frac{\mathbf{x_t}\mathbf{Q_t}^\top \odot \mathbf{x_0}\overline{\mathbf{Q}}_\mathbf{t-1}}{\mathbf{x_0}\overline{\mathbf{Q}}_\mathbf{t}\mathbf{x_t}^\top}\right) \tag{11}$$

As in (Austin et al., 2021b), the neural network responsible for denoising $p_\theta(\tilde{\mathbf{x}}_\mathbf{0}|\mathbf{x_t}, g)$ is trained to predict the original data $\mathbf{x_0}$. During the reverse process, this predicted $\tilde{\mathbf{x}}_\mathbf{0}$ is used as a substitute for $\mathbf{x_0}$ to calculate the posterior distribution:

$$p_\theta(\mathbf{x_{t-1}}|\mathbf{x_t}) = \sum_{\boldsymbol{x}} q(\mathbf{x_{t-1}}|\mathbf{x_t}, \tilde{\mathbf{x}}_\mathbf{0}) p_\theta(\tilde{\mathbf{x}}_\mathbf{0}|\mathbf{x_t}, g) \tag{12}$$

B. Neural Network Architecture

Following Sun & Yang (2023), we also utilize an anisotropic graph neural network with edge gating (Bresson & Laurent, 2018a;b) for backbone network of the diffusion model.

Consider h_i^ℓ and e_{ij}^ℓ as the features of node i and edge ij at layer ℓ, respectively. Additionally, let t represent the sinusoidal features (Vaswani et al., 2017) corresponding to the denoising timestep t. The propagation of features to the subsequent layer is performed using an anisotropic message-passing mechanism:

$$\hat{e}_{ij}^{\ell+1} = P^\ell e_{ij}^\ell + Q^\ell h_i^\ell + R^\ell h_j^\ell, \tag{13}$$

$$e_{ij}^{\ell+1} = e_{ij}^\ell + \mathrm{MLP}_e(\mathrm{BN}(\hat{e}_{ij}^{\ell+1})) + \mathrm{MLP}_t(t), \tag{14}$$

$$h_i^{\ell+1} = h_i^\ell + \alpha(\mathrm{BN}(U^\ell h_i^\ell + \sum_{j\in N_i} \sigma(\hat{e}_{ij}^{\ell+1}) \odot V^\ell h_j)), \tag{15}$$

where $U^\ell, V^\ell, P^\ell, Q^\ell, R^\ell \in \mathbb{R}^{d\times d}$ are learnable parameters for layer ℓ, α denotes the ReLU activation function (Krizhevsky, 2010), BN stands for Batch Normalization (Ioffe & Szegedy, 2015), A signifies the aggregation function implemented as SUM pooling (Xu et al., 2019), σ is the sigmoid activation function, $\odot$ represents the Hadamard product, N_i indicates the neighbors of node i, and $\mathrm{MLP}(\cdot)$ refers to a two-layer multi-layer perceptron.

For the Traveling Salesman Problem (TSP), the initial edge features e_{ij}^0 are derived from the corresponding values in x_t, and the initial node features h_i^0 are initialized using the nodes' sinusoidal features. In contrast, for the Maximum Independent Set (MIS) problem, e_{ij}^0 are initialized to zero, and h_i^0 are assigned values corresponding to x_t. We then apply a classification or regression head, with two neurons for classification and one neuron for regression, to the final embeddings of x_t (i.e., $\{e_{ij}\}$ for edges and $\{h_i\}$ for nodes) for discrete and continuous diffusion models, respectively.

C. Related works

ML-based CO solvers can be divided into two categories based on their training procedures: RL and SL methods. RL methods iteratively refine subsolutions(da Costa et al., 2020; Wu et al., 2019; Chen & Tian, 2019b; Li et al., 2021; Hou et al., 2023) or extend a partial solution until a complete solution is formed (Kool et al., 2019b; Bello et al., 2016; Kwon et al., 2020; Kim et al., 2022), offering the significant advantage of directly optimizing the given objective. However, because the learning process involves exploring and finding good solutions independently without any guidance from the beginning, it is not easy to train on large-scale problems with a vast search space. On the other hand, SL methods (Joshi et al., 2019a; Fu et al., 2021a; Geisler et al., 2022; Joshi et al., 2019b) predict a solution in one step without iterative refinement. This allows for relatively stable training on large-scale problems, thanks to the availability of a training dataset. However, these methods heavily depend on the quality of the training dataset, and because cost information is not inherently considered, the solutions they produce may not be optimal in practice.

Generative models have shown remarkable success in images and texts, leading to various studies proposing their application in CO with the expectation of leveraging their powerful expressiveness(Graikos et al., 2022a; Mirhoseini et al., 2021; Kool et al., 2019a; Niu et al., 2020; Sun & Yang, 2023; Li et al., 2023). Treating the CO solution generation process as image generation, those methods usually utilize probabilistic generative models to train the solver to sample CO solutions. Recently, (Sun & Yang, 2023), which is closely related to our work, proposes diffusion model-based CO solvers called DIFUSCO, and shows the promising results in various CO problems. However, since generative models are mostly trained using SL, those methods also share the same drawbacks as SL methods in CO. To overcome them, (Li et al., 2023) extends DIFUSCO by integrating a cost-guided local search during the denoising process, thereby better aligning with the true goal of CO, finding optimal solutions for individual instances. (Li et al., 2023) is similar to our work in that it additionally utilized cost information, but it requires a differentiable cost function, which is not always straightforward to define in CO problems.