MONTE CARLO TREE DIFFUSION FOR COMBINATORIAL OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion-based methods have recently shown great promise in solving combinatorial optimization (CO) problems, but reaching high solution quality typically requires aligning the denoising process with the optimization objective. Existing inference-time alignment approaches rely primarily on gradient-based guidance. However, such methods are inherently limited: they require differentiable energy functions, which are often unavailable or difficult to construct for general CO problems. We therefore propose Monte Carlo Tree Diffusion for Combinatorial Optimization. Instead of relying on gradient information, our method performs look-ahead planning using value function evaluations that estimate the quality of future denoising outcomes, with a selection temperature controlling how strongly these estimates steer sampling. Treating the temperature as the search action, our method integrates Monte Carlo tree search into the denoising procedure, enabling adaptive exploration and recovery from suboptimal intermediate decisions under high noise levels. Beyond inference-time alignment, we incorporate this alignment method into training through trajectory-level weighting and auxiliary objectives, improving the base diffusion model without additional inference cost. Experiments on four standard CO benchmarks show that our method improves the unsupervised diffusion solver it builds on in every setting tested, outperforms prior unsupervised baselines on TSP, CVRP and STP, and is competitive with supervised approaches; on MIS, where the base solver is already near saturation, the gains are smaller.

1 INTRODUCTION

Combinatorial optimization (CO) problems arise in a wide range of real-world applications, including logistics (Ngoc Bao et al., 2018), scheduling (Smit et al., 2026), resource allocation (Solozabal et al., 2020), and circuit design (Held et al., 2011). Classical exact methods, such as branch-and-bound (Lawler & Wood, 1966), can become computationally expensive as problem size grows, motivating the development of fast and scalable approximate solvers.

Recent advances in machine learning have led to substantial progress in neural CO solvers. Existing approaches include supervised methods that imitate expert solutions or solver decisions (Vinyals et al., 2017; Joshi et al., 2019; Hudson et al., 2022), reinforcement learning methods that directly optimize task-specific objectives (Kwon et al., 2021; Kool et al., 2019; Kim et al., 2023), and, more recently, generative approaches based on diffusion models (Sun & Yang, 2023; Li et al., 2023) and flow matching (Li et al., 2026a). Despite their strong performance, most diffusion-based CO solvers rely heavily on high-quality solutions produced by classical solvers for training. Such supervision can be expensive to obtain precisely in the large-scale or difficult regimes where learned solvers are most appealing. This limitation has motivated growing interest in unsupervised neural CO solvers. DiffUCO (Sanokowski et al., 2025b), for example, introduces an unsupervised diffusion framework for CO, while SDDS (Sanokowski et al., 2025a) further improves its scalability by addressing the memory bottleneck of discrete diffusion sampling.

A central limitation of these unsupervised approaches is that solution quality is constrained by the model's own sampling distribution. This naturally suggests diffusion alignment: steering the reverse process toward higher-quality outcomes (Uehara et al., 2025; 2024).

However, existing approaches typically rely on local reward or gradient signals to approximate the desired aligned sampling distribution. For combinatorial optimization problems, this approximation

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

is particularly challenging because solution quality is often determined by complex global constraints and long-horizon dependencies. As a result, prior methods frequently resort to differentiable surrogates of the objective, such as relaxations defined on the raw diffusion output rather than on the decoded solution, which may only partially reflect true solution quality and can misguide the denoising process. Moreover, directly sampling from the optimal value-aware aligned distribution is generally intractable, making effective exploration–exploitation balancing a central challenge for diffusion alignment in CO.

We address these limitations with *Monte Carlo Tree Diffusion for Combinatorial Optimization*, a derivative-free alignment framework that casts diffusion guidance as sequential decision-making over how strongly to exploit the value function. At each denoising step, we sample candidate states from the pretrained reverse process and reweight them according to a learned outcome network V_ϕ. Their selection is controlled by a temperature α, which determines the degree of greediness: as $\alpha \to \infty$, the procedure approaches unguided reverse-process sampling, whereas as $\alpha \to 0$, it approaches greedy maximization of the predicted outcome. We then treat α itself as the action in a Monte Carlo tree search over denoising trajectories. In this way, the selection temperature serves as a derivative-free analogue of a guidance scale, while tree search introduces look-ahead and backtracking beyond single-trajectory sampling.

Crucially, our tree search does not rely on expensive trajectory rollouts. Instead, V_ϕ directly estimates the outcome associated with a search node, reducing node evaluation to a single network forward pass. This makes tree search sufficiently inexpensive to use not only at inference time, but also inside the training loop. During training, the tree adaptively balances exploration and exploitation when generating diffusion trajectories, and the resulting trajectories are subsequently used for PPO-based policy updates with trajectory-level weighting. Related work in other domains has similarly demonstrated the value of incorporating search or sequential sampling into training-time alignment (Hiranaka et al., 2025; Yang et al., 2024; Macfarlane et al., 2024). In our setting, this training-time use of search is particularly important: it improves the underlying diffusion solver itself, preserving its original inference cost when search is disabled, while retaining test-time tree search as an optional accuracy–latency trade-off.

Our contributions are threefold. (i) We introduce a derivative-free formulation of tree-search diffusion alignment for combinatorial optimization, where search actions control the selection temperature of value-weighted resampling rather than a gradient-guidance scale. (ii) We develop a rollout-free search procedure that evaluates intermediate diffusion states using a learned outcome network, making trajectory-level planning efficient enough to use during training. (iii) We propose a training-time alignment framework that converts tree-guided exploration into a learning signal through trajectory-level policy optimization. Across TSP, CVRP, STP, and MIS, our method consistently improves the unsupervised diffusion solver on which it is built in all eleven evaluated settings, at matched inference cost. It further outperforms prior unsupervised baselines on TSP, CVRP, and STP and achieves competitive performance relative to supervised solvers. We additionally analyze the large-scale MIS regime, where a stronger baseline remains ahead, to characterize the current limitations of our approach.

2 Related Work

2.1 Diffusion Models

Diffusion models (Ho et al., 2020) iteratively transform noise into structured data and have been extended from continuous to discrete domains (Austin et al., 2023; Shi et al., 2025). DIFUSCO (Sun & Yang, 2023) first applied them to CO, followed by masked-diffusion variants (Wang et al., 2026). Flow matching (Lipman et al., 2023; Gat et al., 2024) offers an alternative transport-based formulation, adapted to CO by FMIP (Li et al., 2026a). These solvers are largely supervised: they require large amounts of solver-generated labels, which are prohibitively expensive for complex real-world tasks.

2.2 Unsupervised CO Solvers

A growing line of work therefore solves CO without labels. POMO (Kwon et al., 2021) exploits solution symmetries through diverse rollout starting points; UTSP (Min et al., 2024) learns TSP edge-selection heatmaps without supervision; RLNN (Feng & Yang, 2025) regularises discrete Langevin sampling. Most relevant to us, DiffUCO (Sanokowski et al., 2025b) trains diffusion solvers

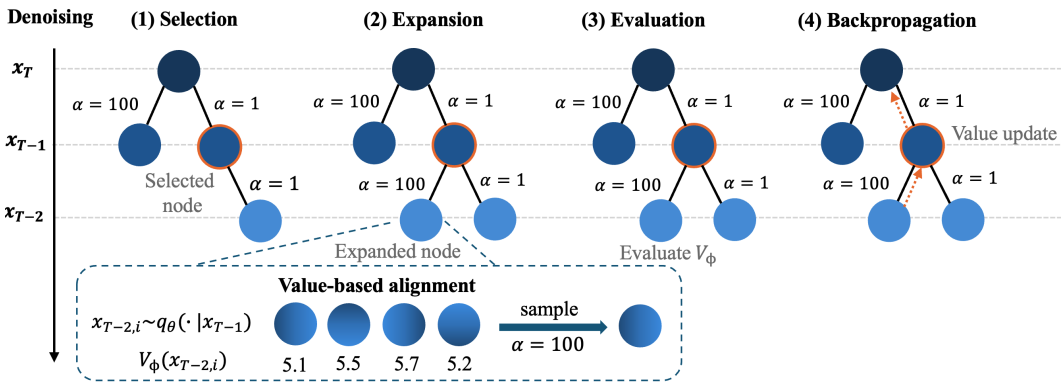

Figure 1: Derivative-free Monte Carlo Tree Diffusion for CO: temperature actions over value-weighted resampling replace gradient-guidance scales, and a learned outcome network V_ϕ replaces rollouts. The same search also generates training trajectories, improving the base solver at unchanged inference cost.

against an energy-based objective, and SDDS (Sanokowski et al., 2025a) scales this framework; our method builds directly on this family.

2.3 Inference-time Diffusion Alignment

Alignment steers a pretrained diffusion model toward high-reward outputs. *Gradient-based guidance* (Dhariwal & Nichol, 2021; Chung et al., 2023; Bansal et al., 2023) modifies the reverse dynamics with value gradients and has been applied to CO (Li et al., 2023; 2024b), but requires differentiable surrogate objectives and provides only local information. *Sequential Monte Carlo* (Doucet et al., 2001; Cardoso et al., 2023; Kim et al., 2025) resamples particle populations toward a reward-weighted target and supports non-differentiable rewards; SVDD (Li et al., 2024a) performs greedy per-step particle selection. *Search-based methods* add look-ahead through beam search (Li et al., 2025), best-of-N with tree search (Zhang et al., 2026), and MCTS over denoising trajectories (Yoon et al., 2026; 2025), with Jain et al. (2025) improving value estimates via reusable rollout statistics and global reward backup. Our method belongs to the search-based family but is derivative-free and, unlike prior work, is used to generate training signal rather than only to refine inference.

3 Preliminaries

3.1 Unsupervised Diffusion Solvers for Combinatorial Optimization

Unlike standard diffusion models (Ho et al., 2020; Sun & Yang, 2023), which learn from a ground-truth data distribution, unsupervised diffusion models instead optimize an energy-based objective. In combinatorial optimization, the target distribution is typically defined as $p_\beta(x) \propto \exp(-\beta H(x;\mathcal{P}))$, where $H(x;\mathcal{P})$ denotes the problem-specific objective function for solution x on instance $\mathcal{P}$, and $\beta = \frac{1}{T}$ is the inverse temperature. The goal of the neural solver $q_\theta(x|\mathcal{P})$ is therefore to generate low-energy solutions by minimizing the expected objective value: $\min_\theta \mathbb{E}_{\mathcal{P},\, x\sim q_\theta(x|\mathcal{P})}\left[H(x;\mathcal{P})\right]$.

Taking the diffusion trajectory into account, unsupervised diffusion models admit a variational objective $D_{KL}(q_\theta(x_{0:T})\|p(x_{0:T}))$, where $q_\theta(x_{0:T})$ denotes the learned reverse diffusion and $p(x_{0:T})$ denotes the target trajectory distribution. Existing methods such as DiffUCO (Sanokowski et al., 2025b) and SDDS (Sanokowski et al., 2025a) optimize this objective through reinforcement learning.

In practice, the learned reverse process deviates from the target distribution through approximation and optimization errors, so generated solutions remain suboptimal — motivating inference-time alignment.

3.2 Diffusion Alignment

Given a pretrained diffusion solver $q_\theta(x_0)$, diffusion alignment aims to steer the sampling process toward higher-quality solutions by sampling from the reward-weighted target distribution: $\tilde{q}(x_0) \propto q_\theta(x_0)\exp(\lambda r(x_0))$, where $r(x_0)$ denotes a reward or solution quality function. The optimal intermediate guidance for this target distribution is characterized by the soft value function

$$V(x_t) = \log \mathbb{E}_{q_\theta(x_0|x_t)}\left[\exp(\lambda r(x_0))\right], \tag{1}$$

which measures the expected future reward conditioned on the current diffusion state x_t. Under this formulation, the optimal aligned reverse dynamics become

$$\tilde{q}(x_{t-1}|x_t) \propto q_\theta(x_{t-1}|x_t)\exp(V(x_{t-1}) - V(x_t)). \tag{2}$$

However, the above optimally aligned intermediate distribution is intractable due to the combinatorial search space, which motivates the design of approximated distributions.

4 Method

4.1 Value-based Diffusion Alignment

Gradient guidance modifies each transition with local gradients such as $\nabla H(x_t; \mathcal{P})$ (Dhariwal & Nichol, 2021); as Section 3.2 shows, this is only a heuristic approximation to the optimal guidance, whose exact form is characterized by the soft value function, and it requires a differentiable objective that CO problems do not provide.

To address these limitations, we adopt a value-based alignment framework in which an *outcome* value network $V_\phi(x_t)$ is trained to predict the expected future solution quality conditioned on the intermediate diffusion state x_t. Throughout, V_ϕ denotes this outcome network and is distinct from the PPO critic V_ψ used for advantage estimation: only V_ϕ guides sampling and scores nodes during search, and the two are trained separately (Appendix A.2). This value function serves as an approximation to the optimal soft value function introduced in Section 3.2.

At each diffusion step, M candidate states $\{x_{t-1}^{(i)}\}_{i=1}^{M}$ are sampled from the reverse transition distribution $q_\theta(x_{t-1}|x_t)$, following Soft Value-Based Decoding in Diffusion models (SVDD) (Li et al., 2024a). Each candidate is assigned a weight

$$w_{t-1}^{(i)} = \exp\left(\frac{V_\phi(x_{t-1}^{(i)})}{\alpha}\right), \tag{3}$$

where α denotes the selection temperature, which controls how sharply the weights concentrate on high-value candidates. We emphasise that α is distinct from the energy temperature $\mathcal{T}$ of Section 3.1: α governs candidate selection at inference, whereas $\mathcal{T}$ defines the target distribution itself. The next diffusion state is then sampled according to

$$i \sim \text{Categorical}\left(\left\{\frac{w_{t-1}^{(i)}}{\sum_{j=1}^{M} w_{t-1}^{(j)}}\right\}_{i=1}^{M}\right). \tag{4}$$

The value network is trained by regressing each intermediate state onto the objective value actually realised by its trajectory. Concretely, given a denoising trajectory $x_T, \ldots, x_0$ sampled under the current policy and its decoded solution x_0', we minimise the mean squared error

$$\mathcal{L}_V = \mathbb{E}_{x_{0:T}}\left[\frac{1}{T}\sum_{t=1}^{T}\left(V_\phi(x_t) - r(x_0')\right)^2\right], \tag{5}$$

where $r(x_0') = -H(x_0'; \mathcal{P})$ is the evaluated solution quality of the decoded trajectory endpoint. Since x_0' is already computed for the reward signal, the value targets are obtained at no additional cost. This Monte Carlo regression estimates $\mathbb{E}_{q_\theta(x_0|x_t)}[r(x_0)]$ rather than the exact soft value of Equation (1); the selection temperature α absorbs the resulting mismatch by controlling how aggressively the estimate is trusted.

Because V_ϕ is imperfect, we sample multiple temperatures at inference to span levels of exploration and exploitation: low α selects greedily toward high-value candidates, high α preserves the stochasticity of the reverse process. The limits recover familiar behaviours — $\alpha \to \infty$ reduces to unguided ancestral sampling and $\alpha \to 0$ to greedy selection — so α plays the role of a guidance scale without requiring a differentiable objective.

4.2 Monte Carlo Tree Diffusion for Combinatorial Optimization

Although value-guided sampling improves local transition quality, it remains fundamentally myopic: candidate selection is performed independently at each denoising step without considering future trajectory quality. To address this limitation, we introduce Monte Carlo Tree Diffusion (MCTD) for combinatorial optimization, illustrated in Figure 1.

We formulate diffusion generation as a planning problem over denoising trajectories. Each intermediate diffusion state x_t is treated as a node in the search tree, while actions correspond to different sampling temperatures α. Instead of searching over gradient guidance scales (Yoon et al., 2026), our framework performs derivative-free search over temperature-controlled sampling policies.

MCTD then iteratively repeats four stages:

Selection. Starting from the root, child nodes are recursively selected by an Upper Confidence Bound criterion balancing exploration and exploitation (Equation 8, Appendix A.2), steering traversal toward promising denoising trajectories.

Expansion. Given the selected node x_t and a temperature action α, we generate candidate transitions from the reverse diffusion model

$$x_{t-1}^{(i)} \sim q_\theta(x_{t-1}|x_t). \tag{6}$$

Candidates are evaluated using the value network V_ϕ, weighted according to the temperature-controlled distribution introduced previously, and resampled to construct new tree nodes.

Evaluation. Instead of performing costly rollout simulations to terminal states, we directly estimate future trajectory quality using the learned value function V_ϕ. This substantially reduces computational overhead while still enabling trajectory-level evaluation.

Backpropagation. The estimated values are propagated backward through the search tree to update node statistics and improve future selection decisions.

Combining value-based alignment with tree search lets MCTD reason over multiple denoising trajectories and recover from poor intermediate decisions that single-trajectory diffusion cannot revisit. We let each tree transition advance the reverse process by a multi-step jump rather than a single step, so search decisions are made at sparse intervals: this reduces computational cost and limits how many noisy intermediate states must be scored by the value network, whose estimates are least reliable early in denoising (Jain et al., 2025).

4.3 Training-Time Alignment

Beyond inference, we integrate MCTD into training to improve the diffusion policy itself: the search generates trajectories whose exploration is shaped by the temperature actions, providing a stronger optimization signal than standard sampling. To exploit them, each trajectory $x_{0:T}$ receives a weight computed from its terminal objective value $H(x_0; \mathcal{P})$, applied to all of its transitions within the PPO objective. This shifts probability mass toward trajectories that end in high-quality solutions while still learning from the full sampled distribution: we deliberately retain suboptimal trajectories rather than filtering them, which preserves diversity and prevents collapse onto a narrow set of solutions. Empirically, trajectory-level weighting combined with MCTD-generated exploration substantially improves both training stability and final solution quality (Section 5.4).

Table 1: Experiment results on TSP datasets

Method	TSP-500			TSP-1000		
	Obj ↓	Gap ↓	Time ↓	Obj ↓	Gap ↓	Time ↓
Concorde (Applegate & Cook, 2006)	16.55	0.00%	37.66m	23.12	0.00%	6.65h
Gurobi (Gurobi Optimization, LLC, 2026)	16.55	0.00%	46.88m	23.12	0.00%	2.57h
GCN (Joshi et al., 2019)	30.37	83.55%	38.02m	51.26	121.73%	51.67m
DIFUSCO (Sun & Yang, 2023)	16.86	1.81%	2.91m	23.91	3.16%	21.10m
FMIP (Li et al., 2026a)	**16.74**	**1.69%**	**2.91m**	23.81	2.98%	21.10m
AM (Kool et al., 2019)	19.53	18.03%	21.99m	29.90	29.23%	1.64m
POMO (Kwon et al., 2021)	19.24	16.25%	12.80h	49.56	114.36%	63.45h
DIMES (Qiu et al., 2022)	17.80	7.55%	2.11h	24.89	7.70%	4.53h
SDDS (Sanokowski et al., 2025a)	17.08	3.13%	2.08m	24.19	4.92%	16.57m
CAM (Feng et al., 2026)	16.91	2.18%	2.91m	23.70	2.51%	21.10m
Ours	16.90	2.10%	2.08m	23.67	2.38%	16.57m
Ours + MCTD sampling	16.81	1.72%	11.57m	**23.63**	**2.21%**	**23.17m**

4.4 Auxiliary Combinatorial Adjoint Matching Objective

To further improve optimization, we incorporate Combinatorial Adjoint Matching (CAM) (Feng et al., 2026) as an auxiliary training objective. CAM propagates terminal gradients backward through the diffusion trajectory, encouraging intermediate states to produce transitions leading to improved final solutions.

For combinatorial optimization problems such as TSP and CVRP, the raw diffusion output x_0 does not necessarily correspond to a feasible solution. Therefore, we decode x_0 into a feasible solution x_0' using task-specific decoding and local search procedures. The auxiliary CAM objective then minimizes the discrepancy between the diffusion prediction and the decoded solution: $\mathcal{L}_{\text{CAM}} = \text{BCE}(x_0, x_0')$, where BCE denotes the binary cross-entropy loss.

This objective introduces negligible additional computational overhead since the decoded solution x_0' is already computed during reward evaluation. By providing additional trajectory-level supervision, CAM further stabilizes training and improves final solution quality.

5 Experiments

All experiments were performed on NVIDIA A6000 GPUs. Test evaluations were run sequentially on a single GPU.

5.1 Datasets

We evaluate our method on four combinatorial optimization problems: the Traveling Salesman Problem (TSP), the Capacitated Vehicle Routing Problem (CVRP), the Steiner Tree Problem (STP), and the Maximum Independent Set (MIS).

TSP. We follow the experimental setup of Sun & Yang (2023). Instances are drawn from the ML4CO benchmark (Ma et al., 2025b). We evaluate on two datasets with 500 and 1000 nodes, respectively, each containing 128 test instances. To obtain valid tours from model outputs, we apply greedy decoding followed by 2-opt local search. The same post-processing pipeline is used for all methods to ensure a fair comparison.

CVRP. Instances are also drawn from the ML4CO benchmark (Ma et al., 2025b). We follow the experimental setup of Ma et al. (2025a). We adopt the Classic-LS post-processing method to obtain feasible routes. The CVRP-100 dataset contains 10,000 test instances, while CVRP-200 and CVRP-500 each contain 100 test instances.

STP. We follow the data generation paradigm of Park et al. (2025) and label these datasets using SCIP-Jack (Rehfeldt & Koch, 2023). We experiment on four Erdős–Rényi (ER) STP datasets, with 10, 20, 30, and 50 nodes in the graphs, respectively. Following Park et al. (2025), each vertex is independently designated a terminal with probability $p_t = 0.2$, so an instance on n nodes has $0.2n$

terminals in expectation. SteBen reports ER, WS, RR and grid instances separately; all STP results here are on the ER family. As in Park et al. (2025), we adopt the decoding strategy of Yan et al. (2021) to obtain feasible solutions.

MIS. Instances are drawn from the ML4CO benchmark (Ma et al., 2025b). We consider Revised Model B (RB) (Xu & Li, 2000) graphs at two scales: small (200–300 nodes) and large (800–1200 nodes). We initialize from pretrained SDDS checkpoints (Sanokowski et al., 2025a) and fine-tune on our setting. For evaluation, we adopt conditional expectation decoding as in SDDS, followed by greedy decoding as in Sun & Yang (2023).

5.2 Baselines and Metrics

Besides the original SDDS (Sanokowski et al., 2025a), we compare with multiple other neural baselines. We mainly compare with the supervised diffusion models (Sun & Yang, 2023). On the CVRP datasets, we further compare with RL4CO (Berto et al., 2025), which learns a next-step prediction policy. On STP, we compare with CherryPick (Yan et al., 2021), which uses deep Q-learning to predict STP solutions. On MIS, we further compare with Intel (Li et al., 2018) and DGL (Böther et al., 2022), two supervised models for solving MIS. We experiment with two variants of our model: one with MCTD used during sampling and another without.

To evaluate a model's performance, we utilize three metrics: the objective value of the problem, the gap compared with the reference results, and the total runtime. The gap is calculated as $\frac{|\text{Obj}_{\text{ref}} - \text{Obj}|}{|\text{Obj}_{\text{ref}}|}$, where Obj_{ref} is the objective value of the reference solution. For STP (Park et al., 2025), the objective values are substantially larger in magnitude, so we report only the optimality gap for readability.

5.3 Experiment Results

Results on TSP datasets are reported in Table 1. The primary comparison is against the solvers we build on, at matched inference cost: our method reaches 2.10% and 2.38% gap in 2.08m and 16.57m, against 3.13%/4.92% for SDDS at exactly the same runtime and 2.18%/2.51% for CAM at a 1.3–1.4$\times$ larger one. Training-time alignment therefore buys accuracy without buying latency, which is the regime that matters when a solver is deployed. On TSP-1000 this is already enough to overtake the supervised diffusion solver DIFUSCO (2.38% vs. 3.16%), despite using no solver-generated labels. Enabling tree search at inference adds a further improvement, at a cost that is highly scale-dependent — $5.6\times$ on TSP-500 but only $1.4\times$ on TSP-1000, since search runs at sparse intervals while the per-step denoising cost grows with instance size. We therefore report it as an optional accuracy/latency knob rather than as the default configuration. Compared with POMO, which needs 12.80h on TSP-500, our method reaches a lower gap in 2.08m.

Results on CVRP datasets are summarized in Table 2. Our method consistently outperforms prior unsupervised baselines, including RL4CO, SDDS, and CAM, across all scales, achieving lower objective values and smaller optimality gaps. In particular, the advantage becomes more pronounced on larger instances, where our method attains a 2.77% gap on CVRP-500 compared with 4.17% for RL4CO and 4.78% for SDDS, suggesting better scalability to harder problems. Furthermore, augmenting our model with MCTD sampling yields additional improvements on every benchmark, reducing the gap to 1.42%, 2.77%, and 2.52% on CVRP-100, CVRP-200, and CVRP-500, respectively. This again demonstrates that stronger test-time search can effectively complement the learned generator by translating additional computation into higher-quality solutions.

Table 3 reports the results on Steiner Tree Problem benchmarks with 10, 20, 30, and 50 nodes. Among neural approaches, our method consistently achieves the best performance on all four datasets. The introduction of MCTD and CAM into training significantly boosts the model performance, reducing the gap by roughly 4.8 percentage points relative to SDDS and by 0.4 points relative to CAM on STP-50. Furthermore, incorporating MCTD sampling yields additional gains on every benchmark, reducing the gap to 0.24%, 0.12%, 0.27%, and 0.21%, respectively.

Table 4 reports results on MIS. This is the one benchmark where the base solver is already close to saturation — SDDS reaches a 1.59% gap on RB-small — leaving little headroom for alignment. Our method still improves on it at both scales (1.59%→0.94% and 5.93%→5.32%), with inference-time search recovering a further 0.39 points on RB-large. It is also the one setting where we do not improve

Table 2: Experiment results on CVRP datasets

Method	CVRP-100			CVRP-200			CVRP-500		
	Obj ↓	Gap ↓	Time ↓	Obj ↓	Gap ↓	Time ↓	Obj ↓	Gap ↓	Time ↓
HGS (Vidal, 2022)	15.56	0.00%	55.56h	19.63	0.00%	1.67h	37.15	0.00%	100h
DIFUSCO (Sun & Yang, 2023)	15.89	2.18%	31.20m	20.25	3.17%	1.11m	38.26	2.99%	8.67m
FMIP (Li et al., 2026a)	15.93	2.40%	31.20m	20.27	3.25%	1.11m	38.30	3.11%	8.11m
COExpander (Ma et al., 2025a)	16.22	4.25%	10.00m	20.59	4.89%	15s	39.12	5.34%	61s
RL4CO (Berto et al., 2025)	15.93	2.38%	28.33m	20.19	2.88%	34s	38.70	4.17%	88s
SDDS (Sanokowski et al., 2025a)	16.04	3.12%	21.67m	20.58	4.82%	52s	38.92	4.78%	7.15m
CAM (Feng et al., 2026)	15.98	2.73%	37.35m	20.38	3.82%	1.21m	38.25	2.96%	8.91m
Ours	15.82	1.71%	21.67m	20.18	2.82%	52s	38.18	2.77%	7.16m
Ours + MCTD sampling	**15.78**	**1.42%**	**31.35m**	**20.15**	**2.77%**	**3.10m**	**38.09**	**2.52%**	**14.22m**

Table 3: Experiment results on Steiner Tree Problem datasets

Method	STP-10		STP-20		STP-30		STP-50	
	Gap ↓	Time ↓	Gap ↓	Time ↓	Gap ↓	Time ↓	Gap ↓	Time ↓
SCIP-Jack (Rehfeldt & Koch, 2023)	0.00%	3.00m	0.00%	3.00m	0.00%	3.00m	0.00%	5.00m
DIFUSCO (Sun & Yang, 2023)	0.43%	7.50m	0.64%	9.26m	1.11%	37.04m	0.74%	1.52h
CherryPick (Yan et al., 2021)	8.42%	13.00m	21.19%	40.00m	20.50%	52.00m	29.22%	1.50h
SDDS (Sanokowski et al., 2025a)	2.71%	5.40m	3.19%	6.67m	4.24%	26.67m	5.11%	1.10h
CAM (Feng et al., 2026)	0.52%	7.50m	0.58%	9.26m	0.91%	37.04m	0.67%	1.52h
Ours	0.33%	5.40m	0.29%	6.67m	0.49%	26.67m	0.27%	1.10h
Ours + MCTD sampling	**0.24%**	**23.00m**	**0.12%**	**22.30m**	**0.27%**	**26.30m**	**0.21%**	**1.80h**

Table 4: Experiment results on MIS datasets

Method	RB-[200-300]			RB-[800-1200]		
	Obj ↑	Gap ↓	Time ↓	Obj ↑	Gap ↓	Time ↓
Gurobi (Gurobi Optimization, LLC, 2026)	19.98	0.60%	47.57m	40.90	5.21%	2.17h
KaMIS (Hespe et al., 2019)	20.10	0.00%	1.40h	43.15	0.00%	2.05h
Intel (Li et al., 2018)	18.47	8.11%	13.07m	34.47	20.12%	20.28m
DGL (Böther et al., 2022)	17.36	13.93%	12.78m	34.50	20.05%	23.90m
DIFUSCO (Sun & Yang, 2023)	18.34	8.76%	24.45m	38.67	10.38%	3.88h
SDDS (Sanokowski et al., 2025a)	19.78	1.59%	2.33m	40.59	5.93%	4.52m
CAM (Feng et al., 2026)	19.91	0.94%	2.00m	**41.25**	**4.40%**	**4.02m**
Ours	19.91	0.94%	2.33m	40.86	5.32%	4.52m
Ours + MCTD sampling	**19.97**	**0.69%**	**5.71m**	40.98	4.93%	10.50m

on CAM, which attains 4.40% on RB-large against our 5.32% and is indistinguishable from us on RB-small. We attribute this to saturation rather than to a failure of search: with the policy already near its achievable ceiling, the value function has little residual signal and the trajectory reweighting that drives our gains elsewhere has little to reweight. MIS is also the only benchmark fine-tuned from a released checkpoint rather than trained from scratch.

5.4 Ablation Studies

We ablate each component we introduce, on the TSP datasets. The results are shown in Table 5.

Effect of CAM. Augmenting the SDDS objective with CAM yields substantial improvements, reducing the gap from 3.13% to 2.26% on TSP-500 and from 4.92% to 2.70% on TSP-1000. This confirms that CAM provides a stronger optimization signal for diffusion training.

Effect of MCTD. Combining SDDS and CAM alone does not reach the standalone CAM solver (2.26% vs. 2.18% on TSP-500). Adding MCTD-generated training trajectories closes and reverses this: our complete method reaches gaps of 2.10% and 2.38% on TSP-500 and TSP-1000, improving on SDDS + CAM in both settings, which indicates that the search supplies higher-quality training trajectories than standard sampling.

Effect of trajectory weighting. Removing the weighting mechanism degrades both settings, with the gap rising to 2.43% and 2.62%. Replacing weighting with quality-based *sampling* of trajectories

Table 5: Ablation studies on TSP datasets

Method	TSP-500			TSP-1000		
	Obj ↓	Gap ↓	Time ↓	Obj ↓	Gap ↓	Time ↓
CAM (Feng et al., 2026)	16.91	2.18%	2.91m	23.70	2.51%	21.10m
SDDS (Sanokowski et al., 2025a)	17.08	3.13%	2.08m	24.19	4.92%	16.57m
SDDS + CAM	16.93	2.26%	2.08m	23.74	2.70%	16.57m
SDDS + SVDD training	17.21	4.03%	2.08m	24.37	5.44%	16.57m
SDDS + SVDD sampling	17.01	3.09%	6.80m	24.16	4.87%	18.39m
SDDS + MCTD sampling	16.97	2.53%	11.57m	24.02	3.89%	23.17m
Ours	16.90	2.10%	2.08m	23.67	2.38%	16.57m
Ours w/o CAM	16.97	2.32%	2.08m	23.78	2.88%	16.57m
Ours w/o weighting	16.99	2.43%	2.08m	23.72	2.62%	16.57m
Ours w/ weighted sampling	16.96	2.51%	2.08m	23.73	2.67%	16.57m
Ours extended sampling	16.90	2.02%	11.03m	23.66	2.30%	21.02m
Ours + MCTD sampling	16.81	1.72%	11.57m	23.63	2.21%	23.17m

("Ours w/ weighted sampling") is worse still: sampling biases the training distribution away from low-quality trajectories, so the model never learns to refine suboptimal states, whereas weighting emphasizes high-quality samples while keeping every trajectory in the distribution.

Effect of SVDD. Applying SVDD directly during both training and inference degrades performance and eventually collapses training. We attribute this to the absence of an exploration–exploitation mechanism: the model overfits to a narrow set of high-reward samples, reducing diversity and destabilizing optimization, mirroring the weighted-sampling result above. MCTD instead balances the two through tree search and is consistently better when applied during SDDS sampling, at a matched number of generated samples. SVDD remains a usable inference-time refinement strategy, but its purely local selection is weaker than structured look-ahead.

Effect of inference budget. Finally, we compare MCTD sampling against simply spending the same compute on more denoising: "Ours extended sampling" increases the number of sampling steps of our trained model until its inference time matches that of MCTD sampling. Extra steps do help, but MCTD remains better at an equal budget, indicating that its gains come from structured, search-based alignment rather than from increased sampling effort alone.

In general, CAM, MCTD, and trajectory weighting are critical components. Their combination yields consistent improvements over baselines, demonstrating the effectiveness of integrating value-based alignment with trajectory-level planning.

6 Conclusion

In this paper, we propose Monte Carlo Tree Diffusion for Combinatorial Optimization. By combining value-based evaluation with tree-structured search, our method replaces gradient-based guidance with a planning-based alignment mechanism that operates on denoising trajectories. Beyond inference-time alignment, we extend our method to training via trajectory-level weighting and auxiliary objectives, leading to more stable and effective learning. Experiments on multiple CO benchmarks demonstrate that our approach improves solution quality over the base solver in every setting tested and outperforms prior unsupervised baselines on TSP, CVRP and STP, while remaining competitive with supervised ones. Applying the search at test time further improves performance at the cost of additional inference computation.

Overall, our results suggest that value-based planning and structured search over generative trajectories provide a promising alternative to conventional local guidance methods for diffusion alignment. More broadly, we believe this perspective opens a new direction toward integrating sequential decision-making and probabilistic planning into generative optimization frameworks.

Limitations. Our gains depend on the base solver having exploitable headroom: on MIS, where the pretrained solver is near saturation, reweighting has little to act on and a stronger baseline remains ahead. The method is also restricted to integer programs; continuous variables would require a mixed diffusion model and a more carefully designed energy function. Finally, tree search is not fully parallelizable, which limits its application to instances with millions of variables.

AI USE STATEMENT

In this work, we used generative AI tools to assist with writing, with implementing methods, with interpreting results, and with summarizing existing literature. We did not use generative AI tools for generating synthetic datasets, developing theoretical models or conceptual frameworks, formulating mathematical claims or providing critical ingredients for proving them, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, assisting with translation, cleaning or reformatting datasets, or supporting qualitative and thematic data analysis. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

David Applegate and William Cook. Concorde tsp solver, 2006.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces, 2023. URL https://arxiv.org/abs/2107.03006.

Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal guidance for diffusion models, 2023. URL https://arxiv.org/abs/2302.07121.

Federico Berto, Chuanbo Hua, Junyoung Park, Laurin Luttmann, Yining Ma, Fanchen Bu, Jiarui Wang, Haoran Ye, Minsu Kim, Sanghyeok Choi, Nayeli Gast Zepeda, André Hottung, Jianan Zhou, Jieyi Bi, Yu Hu, Fei Liu, Hyeonah Kim, Jiwoo Son, Haeyeon Kim, Davide Angioni, Wouter Kool, Zhiguang Cao, Qingfu Zhang, Joungho Kim, Jie Zhang, Kijung Shin, Cathy Wu, Sungsoo Ahn, Guojie Song, Changhyun Kwon, Kevin Tierney, Lin Xie, and Jinkyoo Park. Rl4co: An extensive reinforcement learning for combinatorial optimization benchmark. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '25, pp. 5278–5289. ACM, August 2025. doi: 10.1145/3711896.3737433. URL http://dx.doi.org/10.1145/3711896.3737433.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Yash Katariya, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

Maximilian Böther, Otto Kißig, Martin Taraz, Sarel Cohen, Karen Seidel, and Tobias Friedrich. What's wrong with deep learning in tree search for combinatorial optimization, 2022. URL https://arxiv.org/abs/2201.10494.

Gabriel Cardoso, Yazid Janati El Idrissi, Sylvain Le Corff, and Eric Moulines. Monte carlo guided diffusion for bayesian linear inverse problems, 2023. URL https://arxiv.org/abs/2308.07983.

Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=OnD9zGAGT0k.

Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021. URL https://arxiv.org/abs/2105.05233.

Arnaud Doucet, Nando De Freitas, and Neil Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pp. 3–14. Springer, 2001.

Shengyu Feng and Yiming Yang. Regularized langevin dynamics for combinatorial optimization, 2025. URL https://arxiv.org/abs/2502.00277.

Shengyu Feng, Tarun Suresh, and Yiming Yang. Unsupervised diffusion solver for combinatorial optimization via combinatorial adjoint matching, 2026. URL https://arxiv.org/abs/2605.30920.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching, 2024. URL https://arxiv.org/abs/2407.15595.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2026. URL https://www.gurobi.com.

Stephan Held, Bernhard Korte, Dieter Rautenbach, and Jens Vygen. Combinatorial optimization in vlsi design. *Combinatorial Optimization*, pp. 33–96, 2011.

Demian Hespe, Christian Schulz, and Darren Strash. Scalable kernelization for maximum independent sets. *ACM Journal of Experimental Algorithmics*, 24(1):1.16:1–1.16:22, 2019. doi: 10.1145/3355502. URL https://doi.org/10.1145/3355502.

Ayano Hiranaka, Shang-Fu Chen, Chieh-Hsin Lai, Dongjun Kim, Naoki Murata, Takashi Shibuya, Wei-Hsiang Liao, Shao-Hua Sun, and Yuki Mitsufuji. HERO: Human-feedback efficient reinforcement learning for online diffusion model finetuning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=yMHe9SRvxk.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

Benjamin Hudson, Qingbiao Li, Matthew Malencia, and Amanda Prorok. Graph neural network guided local search for the traveling salesperson problem, 2022. URL https://arxiv.org/abs/2110.05291.

Vineet Jain, Kusha Sareen, Mohammad Pedramfar, and Siamak Ravanbakhsh. Diffusion tree sampling: Scalable inference-time alignment of diffusion models, 2025. URL https://arxiv.org/abs/2506.20701.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL https://kellerjordan.github.io/posts/muon/.

Chaitanya K. Joshi, Thomas Laurent, and Xavier Bresson. An efficient graph convolutional network technique for the travelling salesman problem, 2019. URL https://arxiv.org/abs/1906.01227.

Minsu Kim, Junyoung Park, and Jinkyoo Park. Sym-nco: Leveraging symmetricity for neural combinatorial optimization, 2023. URL https://arxiv.org/abs/2205.13209.

Sunwoo Kim, Minkyu Kim, and Dongmin Park. Test-time alignment of diffusion models without reward over-optimization, 2025. URL https://arxiv.org/abs/2501.05803.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems!, 2019. URL https://arxiv.org/abs/1803.08475.

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning, 2021. URL https://arxiv.org/abs/2010.16011.

Eugene L Lawler and David E Wood. Branch-and-bound methods: A survey. *Operations research*, 14(4):699–719, 1966.

Hongpei Li, Hui Yuan, Han Zhang, Jianghao Lin, Dongdong Ge, Mengdi Wang, and Yinyu Ye. Fmip: Joint continuous-integer flow for mixed-integer linear programming, 2026a. URL https://arxiv.org/abs/2507.23390.

Xiner Li, Yulai Zhao, Chenyu Wang, Gabriele Scalia, Gokcen Eraslan, Surag Nair, Tommaso Biancalani, Shuiwang Ji, Aviv Regev, Sergey Levine, and Masatoshi Uehara. Derivative-free guidance in continuous and discrete diffusion models with soft value-based decoding, 2024a. URL https://arxiv.org/abs/2408.08252.

Xiner Li, Masatoshi Uehara, Xingyu Su, Gabriele Scalia, Tommaso Biancalani, Aviv Regev, Sergey Levine, and Shuiwang Ji. Dynamic search for inference-time alignment in diffusion models, 2025. URL https://arxiv.org/abs/2503.02039.

Yang Li, Jinpei Guo, Runzhong Wang, and Junchi Yan. From distribution learning in training to gradient search in testing for combinatorial optimization. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=JtF0ugNMv2.

Yang Li, Jinpei Guo, Runzhong Wang, Hongyuan Zha, and Junchi Yan. Fast t2t: Optimization consistency speeds up diffusion-based training-to-testing solving for combinatorial optimization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024b. URL https://openreview.net/forum?id=xDrKZOZEOc.

Yang Li, Lvda Chen, Haonan Wang, Runzhong Wang, and Junchi Yan. Generation as search operator for test-time scaling of diffusion-based combinatorial optimization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026b. URL https://openreview.net/forum?id=9JM03CQwzC.

Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Combinatorial optimization with graph convolutional networks and guided tree search, 2018. URL https://arxiv.org/abs/1810.10659.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

Jiale Ma, Wenzheng Pan, Yang Li, and Junchi Yan. COExpander: Adaptive solution expansion for combinatorial optimization. In *Forty-second International Conference on Machine Learning*, 2025a. URL https://openreview.net/forum?id=KMaBXMWsBM.

Jiale Ma, Wenzheng Pan, Yang Li, and Junchi Yan. Ml4co-bench-101: Benchmark machine learning for classic combinatorial problems on graphs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025b. URL https://openreview.net/forum?id=ye4ntB1Kzi.

Matthew V Macfarlane, Edan Toledo, Donal Byrne, Paul Duckworth, and Alexandre Laterre. Spo: Sequential monte carlo policy optimisation, 2024. URL https://arxiv.org/abs/2402.07963.

Yimeng Min, Yiwei Bai, and Carla P. Gomes. Unsupervised learning for solving the travelling salesman problem, 2024. URL https://arxiv.org/abs/2303.10538.

Long Le Ngoc Bao, Duc Hanh Le, and Duy Anh Nguyen. Application of combinatorial optimization in logistics. In *2018 4th International Conference on Green Technology and Sustainable Development (GTSD)*, pp. 329–334, 2018. doi: 10.1109/GTSD.2018.8595447.

Youngjoon Park, Han-Seul Jeong, Kyunghyun Lee, Sungdong Yoo, Hyungseok Song, and Woohyung Lim. Steben: Steiner tree problem benchmark for neural combinatorial optimization on graphs, 2025. URL https://openreview.net/forum?id=tKif2rXQ6V.

Ruizhong Qiu, Zhiqing Sun, and Yiming Yang. Dimes: A differentiable meta solver for combinatorial optimization problems, 2022. URL https://arxiv.org/abs/2210.04123.

Daniel Rehfeldt and Thorsten Koch. Implications, conflicts, and reductions for steiner trees. *Mathematical Programming*, 197:903 – 966, 2023. doi: 10.1007/s10107-021-01757-5.

Sebastian Sanokowski, Wilhelm Berghammer, Martin Ennemoser, Haoyu Peter Wang, Sepp Hochreiter, and Sebastian Lehner. Scalable discrete diffusion samplers: Combinatorial optimization and statistical physics, 2025a. URL https://arxiv.org/abs/2502.08696.

Sebastian Sanokowski, Sepp Hochreiter, and Sebastian Lehner. A diffusion model framework for unsupervised neural combinatorial optimization, 2025b. URL https://arxiv.org/abs/2406.01661.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data, 2025. URL https://arxiv.org/abs/2406.04329.

Igor G. Smit, Yaoxin Wu, Pavel Troubil, Yingqian Zhang, and Wim P.M. Nuijten. Neural multi-objective combinatorial optimization for flexible job shop scheduling problems. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=YAgOaYedLQ.

Ruben Solozabal, Josu Ceberio, and Martin Takáč. Constrained combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:2006.11984*, 2020.

Zhiqing Sun and Yiming Yang. Difusco: Graph-based diffusion solvers for combinatorial optimization, 2023. URL https://arxiv.org/abs/2302.08224.

Masatoshi Uehara, Yulai Zhao, Tommaso Biancalani, and Sergey Levine. Understanding reinforcement learning-based fine-tuning of diffusion models: A tutorial and review, 2024. URL https://arxiv.org/abs/2407.13734.

Masatoshi Uehara, Yulai Zhao, Chenyu Wang, Xiner Li, Aviv Regev, Sergey Levine, and Tommaso Biancalani. Inference-time alignment in diffusion models with reward-guided generation: Tutorial and review, 2025. URL https://arxiv.org/abs/2501.09685.

Thibaut Vidal. Hybrid genetic search for the cvrp: Open-source implementation and swap* neighborhood. *Computers & Operations Research*, 140:105643, April 2022. ISSN 0305-0548. doi: 10.1016/j.cor.2021.105643. URL http://dx.doi.org/10.1016/j.cor.2021.105643.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks, 2017. URL https://arxiv.org/abs/1506.03134.

Yu Wang, Yang Li, Junchi Yan, and Yi Chang. StrudiCO: Structured denoising diffusion with gradient-free inference-stage boosting for memory and time efficient combinatorial optimization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=P69X3V4WwH.

Ke Xu and Wei Li. Exact phase transitions in random constraint satisfaction problems, 2000. URL https://arxiv.org/abs/cs/0004005.

Zong Yan, Haizhou Du, Jiahao Zhang, and Guoqing Li. Cherrypick: Solving the steiner tree problem in graphs using deep reinforcement learning. In *2021 IEEE 16th Conference on Industrial Electronics and Applications (ICIEA)*, pp. 35–40, 2021. doi: 10.1109/ICIEA51954.2021.9516291.

Kai Yang, Jian Tao, Jiafei Lyu, Chunjiang Ge, Jiaxin Chen, Qimai Li, Weihan Shen, Xiaolong Zhu, and Xiu Li. Using human feedback to fine-tune diffusion models without any reward model, 2024. URL https://arxiv.org/abs/2311.13231.

Jaesik Yoon, Hyeonseo Cho, Yoshua Bengio, and Sungjin Ahn. Fast monte carlo tree diffusion: 100x speedup via parallel sparse planning, 2025. URL https://arxiv.org/abs/2506.09498.

Jaesik Yoon, Hyeonseo Cho, Doojin Baek, Yoshua Bengio, and Sungjin Ahn. Monte carlo tree diffusion for system 2 planning, 2026. URL https://arxiv.org/abs/2502.07202.

Tao Zhang, Jia-Shu Pan, Ruiqi Feng, and Tailin Wu. Vfscale: Intrinsic reasoning through verifier-free test-time scalable diffusion model, 2026. URL https://arxiv.org/abs/2502.01989.

A Implementation Details

A.1 Network Architecture Details

TSP. We employ a 12-layer Anisotropic Graph Neural Network with hidden dimension 256 as in DIFUSCO (Sun & Yang, 2023). It is noteworthy that we adopt a different TSP representation scheme from the TSP implementation in DiffUCO. Although DiffUCO provides a node-based representation of TSP, which has a QUBO energy form, this formulation is not scalable. In DIFUSCO's edge representation, we learn whether to include an edge in the final route. For a node representation, we are trying to directly learn the route and the corresponding city order along the route. However, for larger-scale TSP and for edge-based representation, we will construct a sparse graph to rule out those unlikely paths. This cannot be done for this node representation since we do not pose any constraints on the city ordering. For a more scalable neural TSP solver, we adopt the edge representation for our diffusion solver. We draw 4 samples per instance for the TSP datasets. We use 10 steps for inference and 16 runs for DIFUSCO, FMIP, and CAM baselines.

CVRP. We also employ a 12-layer Anisotropic Graph Neural Network with a hidden dimension of 256. Compared to TSP, the model must additionally encode CVRP-specific information, such as customer demand and the depot. We follow settings similar to those of Kool et al. (2019). The node encoding is augmented with customer demand information and a depot indicator, allowing the model to distinguish the depot from regular customers and reason about load allocation. Edge features are also enriched with pairwise distance information in addition to the current route state. All other settings are identical to the model used for TSP. For the CVRP datasets, we evaluate with a single sample to match the other baselines. We use 10 steps for inference and 16 runs for DIFUSCO, FMIP, and CAM baselines.

STP. We also utilize a 12-layer Anisotropic Graph Neural Network for encoding. However, following Park et al. (2025), we modify the edge embedding initialization process to provide the crucial edge costs and terminal node information for Steiner tree problems. Specifically,

$$e_{ij}^0 = W^0[f_\theta(x_t), W^c x_{cost}, W^i x_{ind}] \tag{7}$$

where x_{cost} is the edge cost matrix, x_{ind} is the edge indicator matrix, and f_θ is the embedding method utilized in the original graph neural network. This additional information helps the model capture problem-specific features and improves performance. We draw a single sample per instance for the STP datasets. Following Park et al. (2025), the DIFUSCO and CAM baselines use 50 diffusion steps for sampling on STP.

MIS. We directly adopt the same model settings as in DiffUCO (Sanokowski et al., 2025b;a) since we directly perform finetuning on the provided checkpoints. We draw 4 samples per instance for the MIS datasets.

Our framework uses two separate value networks, which we keep notationally distinct throughout. The first is the standard PPO critic V_ψ, used only for advantage estimation during policy optimization. The second is the outcome network V_ϕ, which predicts the objective value an intermediate diffusion state will ultimately attain; this is the network that guides resampling and scores nodes during tree search. For V_ϕ we adopt the same architecture as the solution network, replacing the original head with a value head that outputs a scalar prediction for each diffusion state. For the MIS problem, before starting MCTD training, we first pretrain V_ϕ for 5 epochs to enable better and more accurate guidance for an already good model. For all other benchmarks, V_ϕ is trained jointly with the policy network, though with a separate optimizer for ease of implementation. This also prevents value-network training from interfering with the original optimization objective.

A.2 Search Procedure and Pseudocode

Selection criterion. Let a node of the search tree be an intermediate diffusion state x, and let an action be a selection temperature $\alpha \in \mathcal{A}$. Writing $N(x, \alpha)$ for the visit count of the child reached from x by action α and $\bar{V}(x, \alpha)$ for that child's current value estimate (maintained by the backpropagation rule below), selection descends the tree by repeatedly choosing

$$\alpha^\star = \arg\max_{\alpha \in \mathcal{A}} \left[\bar{V}(x, \alpha) + c \sqrt{\frac{\ln\left(\sum_{\alpha'} N(x, \alpha') + 1\right)}{N(x, \alpha)}} \right], \tag{8}$$

where $c = \sqrt{2}$ is the standard UCT exploration constant (small constants stabilise both the logarithm and the denominator in the implementation). The first term exploits actions whose sub-trees have produced high-value trajectories; the second favours temperatures that have been tried less often at this node. Because the action set is a temperature ladder rather than a set of gradient scales, Equation (8) requires no differentiation of the objective.

Value backpropagation. Each expansion creates a single child, whose value is estimated by one forward pass of V_ϕ rather than by a rollout to x_0. The estimate v is then propagated along the path back to the root: every ancestor x updates its running mean $V(x) \leftarrow \big(N(x)V(x) + v\big)/\big(N(x) + 1\big)$, increments its visit count, and refreshes the child statistics $\bar{V}, N$ used by Equation (8). Algorithms 1 and 2 summarise inference and training.

Algorithm 1 Derivative-free MCTD sampling

Require: denoiser q_θ, outcome network V_ϕ, instance $\mathcal{P}$, diffusion steps T, temperature set $\mathcal{A}$ with per-action jump lengths $\{k_\alpha\}$, candidates per expansion M, search budget S, leaf temperature τ, number of solutions n

initialise the tree with root $x_T \sim$ prior; $N(\cdot) \leftarrow 0$, $V(\cdot) \leftarrow 0$
for $s = 1, \ldots, S$ **do** ▷ terminates early once every path is fully expanded
  Select: descend from the root via Equation (8) until reaching a node x_t whose selected child is unexpanded
  Expand: under temperature action α, advance the reverse process by a jumpy transition of k_α denoising steps (Yoon et al., 2026) to draw M candidates $\{x^{(i)}_{t-k_\alpha}\}_{i=1}^{M}$; weight $w^{(i)} = \exp(V_\phi(x^{(i)}_{t-k_\alpha})/\alpha)$ and resample one candidate as the new child
  Evaluate: $v \leftarrow V_\phi(\text{child})$ ▷ single forward pass; no rollout to x_0
  Backpropagate: for the child and each of its ancestors x: $V(x) \leftarrow \frac{N(x)V(x)+v}{N(x)+1}$, $N(x) \leftarrow N(x) + 1$
end for
$\mathcal{L} \leftarrow$ fully denoised leaves of the tree; sample n leaves $\propto \mathrm{softmax}\big(V(\ell)/\tau\big)_{\ell \in \mathcal{L}}$
decode each sampled leaf to a feasible solution x'_0
return the best decoded solution

Algorithm 2 Training with search-enhanced trajectories

Require: initial policy q_θ, outcome network V_ϕ, PPO critic V_ψ, CAM coefficient $\lambda_{\text{CAM}} = 1$

for each training iteration **do**
  collect trajectories $\{x^{(n)}_{0:T}\}$ with Algorithm 1
  decode each endpoint to $x'^{(n)}_0$ and evaluate $r(x'^{(n)}_0) = -H(x'^{(n)}_0; \mathcal{P})$
  assign each trajectory a scalar weight from its terminal objective value; broadcast it to all transitions of that trajectory ▷ low-quality trajectories are retained, not filtered
  $\theta \leftarrow \theta - \eta \nabla_\theta \big[\mathcal{L}^{\text{weighted}}_{\text{PPO}} + \lambda_{\text{CAM}} \mathcal{L}_{\text{CAM}}\big]$
  $\psi \leftarrow \psi - \eta_\psi \nabla_\psi \mathcal{L}_{\text{critic}}$ ▷ PPO critic
  $\phi \leftarrow \phi - \eta_V \nabla_\phi \mathcal{L}_V$ ▷ Equation (5); separate optimizer, no shared parameters
end for

A.3 Alignment Implementation Details

Following Sanokowski et al. (2025b), the code is implemented in JAX (Bradbury et al., 2018). Hence, we reimplement MCTD (Yoon et al., 2026) in JAX. As for hyperparameter selection, in the value-based alignment (SVDD) we sample four sub-trajectories per MCTD step. We follow Sanokowski et al. (2025a) and use 12 diffusion steps for all models. We perform the inference with 4 samples for vanilla inference. We also follow Sanokowski et al. (2025b) and perform 3 evaluation steps per diffusion step on all datasets. We use an interval step of 6 for most models. The temperature action space is $\{100, 1, 0.8, 0.1\}$, so that the model performs both exploration ($\alpha = 100$) and exploitation ($\alpha = 0.1$). All other MCTD-related parameters are set to their default values from the original paper.

A.4 Training Details

MIS training generally follows the original settings of Sanokowski et al. (2025a). However, since we fine-tune from a pretrained checkpoint, we reduce the learning rate. On RB-small, we use a learning rate of 5×10^{-4} as opposed to the original 1×10^{-3}. For RB-large, we downscale from 5×10^{-4} to 5×10^{-5}.

For the other three datasets, we adopt settings similar to those of Li et al. (2026b). We use the Muon optimizer (Jordan et al., 2024) with the default learning rate and weight decay of Li et al. (2026b).

For the training loss, the CAM loss is added to the original PPO loss with a coefficient of 1.