MESCO: LEARNING TO SOLVE LARGE-SCALE TSP WITH SELF-SUPERVISION VIA MCTS ENSEMBLES

Anonymous authors
Paper under double-blind review

ABSTRACT

Many neural methods for solving the Traveling Salesman Problem (TSP) follow the heatmap-guided Monte Carlo Tree Search (MCTS) paradigm, in which a neural network assigns a probability to each edge and MCTS uses those probabilities to guide an improvement search. The strongest of such methods train on solved TSP instances, but at scale, this supervision becomes a bottleneck: exact solvers are intractable beyond a few thousand nodes, and even heuristic solutions cost over an hour per instance at 10,000 nodes. This cost of supervision, rather than model capacity, appears to be the limit, as a recent study found that at 10,000 nodes, a simple heuristic heatmap beats every neural method, including supervised methods. To address the challenge of scale, we introduce MESCO, which derives supervision from an ensemble of short MCTS rollouts run during training. Each rollout follows the model's current heatmap, samples its own search configuration, and runs for a fixed budget instead of a solve time dictated by the instance. Aggregating the edges selected by the rollouts, weighted by the quality of the solutions they produce, yields a heatmap target that is more informative than any individual rollout's tour, and improves as the model does. To our knowledge, MESCO is the first neural heatmap method to place MCTS inside the training loop. MESCO attains the best reported optimality gaps on TSP problems with 1,000 and 10,000 nodes, 0.51% and 1.10%, and is competitive with supervised methods at 500 nodes with a gap of 0.22%, while reducing the cost of supervision by up to $192\times$.

1 INTRODUCTION

The Traveling Salesman Problem (TSP) is a fundamental combinatorial optimization problem, with applications to real-world problems such as vehicle routing and warehouse management (Laporte, 1992; Pop et al., 2024). Classical solvers such as Concorde (Applegate et al., 2006) solve TSP instances exactly, but because TSP is NP-hard, obtaining exact solutions becomes intractable for problem sizes beyond a few thousand nodes. Approximation-based methods and metaheuristics such as LKH-3 (Helsgaun, 2017) scale far better and remain state-of-the-art in industrial practice. Neural combinatorial optimization (NCO) (Bello et al., 2017; Wu et al., 2025) pursues a complementary direction, learning problem structure from data rather than encoding it by hand. NCO amortizes the cost of solving TSP instances into a training phase: once trained, an NCO model produces solutions for new instances quickly, without re-deriving a search strategy for each instance.

For solving large-scale TSP, the heatmap-guided Monte Carlo Tree Search (MCTS) paradigm (Fu et al., 2021; Xia et al., 2024; Pan et al., 2026) has recently become a popular choice among NCO methods. In this paradigm, a neural network outputs a probability for each edge of a problem instance, forming a *heatmap*, and MCTS decodes those probabilities to initialize a tour and to bias the k-opt swaps that improve the tour. Fu et al. (2021) introduced this paradigm with Att-GCN, an attention-based graph neural network trained on small TSP instances of 20 and 50 nodes. Since then, supervised discrete denoising diffusion models have become the superior method in this paradigm. DIFUSCO (Sun & Yang, 2024) and Fast-T2T (Li et al., 2024) refine edge probabilities over successive denoising steps, and both are trained by adding noise to a known tour and learning to predict the noise. This scheme requires a known solution for every training example, and since exact solvers cannot provide them beyond a few hundred nodes, both methods rely on LKH-3 instead.

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.

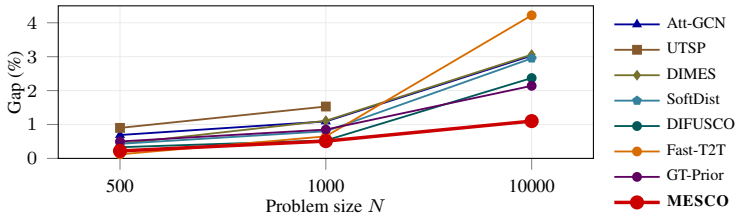


Figure 1: **Optimality gap across different scales of TSP.** On TSP-10000, the best prior method is GT-Prior, a parameter-free distance prior, at 2.14%. Our method, MESCO, attains 1.10% on TSP-10000. Baseline gaps are from Pan et al. (2026); UTSP does not report results on TSP-10000.

Although supervised diffusion models have succeeded on small problem sizes, their success does not persist at scale. Pan et al. (2026) introduced GT-Prior, a simple heuristic heatmap that assigns a fixed empirical probability to each node's k-th nearest neighbors. Re-evaluating every heatmap generator under hyperparameter-tuned MCTS, they find that GT-Prior attains the best optimality gap in the paradigm at TSP-10000, ahead of supervised diffusion models. Figure 1 shows this trend in detail. At 500 nodes, DIFUSCO and Fast-T2T achieve strong performance, but as the problem size increases to 1,000 and 10,000 nodes, their performance degrades significantly. At 10,000 nodes, both Fast-T2T and DIFUSCO perform worse than GT-Prior, a method with no learned parameters.

The cost of supervision plays a key role in this degradation. LKH-3 requires 79 minutes to solve a single TSP-10000 instance on our hardware, so DIFUSCO's $6{,}400$-instance training set costs roughly $8{,}500$ CPU-hours before training begins. For that investment, DIFUSCO attains a 2.37% gap and still lags behind GT-Prior. Label-free methods avoid this cost entirely: DIMES (Qiu et al., 2022) uses an RL meta-learning objective and UTSP (Min et al., 2023) uses a hand-designed surrogate loss, but they are both outperformed by SoftDist, a simple heuristic heatmap based on the softmax over an instance's distance matrix (Xia et al., 2024). Neither supervised diffusion nor its label-free alternatives produce high-quality heatmaps at the scales where NCO methods are most desirable.

A diffusion framework for TSP that needs no solutions. We first address the requirement that makes supervised diffusion expensive. DIFUSCO and Fast-T2T train standard discrete diffusion models, starting the forward process from a known solution, adding noise, and learning to reverse the noise. Their per-step objective depends on a posterior that is available in closed form only because that solution is known and feasible. We propose a framework that does not assume access to known solutions and starts from the noisy solutions themselves. Unlike previous works, we run the full denoising process on a noisy solution conditioned on a TSP problem instance, read the heatmap from the raw probabilities of the final denoising step, and supervise those probabilities directly, comparing sampled heatmaps to heatmap targets. Therefore, our proposed framework is self-supervised, where the supervision is generated from the heatmap output of the denoising process directly.

Self-supervision from MCTS ensembles. Furthermore, we propose a heatmap target generated from MCTS. Using many MCTS rollouts, we form *MCTS ensembles* that, in aggregate, discover the optimal edges needed to form informative heatmap targets. Each search has a random configuration and is seeded by the heatmap the model currently predicts, so the tours they return are individually sub-optimal but explore the solution space differently. We then treat each rollout's choice of successor for a node as a vote, weighting votes by the length of the tour they came from. The resulting target assigns high probability to optimal edges that no individual rollout found. Because every rollout is seeded by the model's own prediction, refreshing the targets periodically closes a loop in which a sharper heatmap yields better-guided searches and those searches yield sharper targets. To our knowledge, this is the first neural heatmap-guided MCTS solver to place MCTS inside the training loop rather than treating it purely as a decoder. Our proposed framework, MESCO, attains the best reported gaps on TSP-1000 and TSP-10000, while reducing the cost of supervision by up to $192\times$.

2 Related Work

NCO methods for TSP can be divided into construction heuristics, which either learn to build tours autoregressively or output edge probabilities in parallel non-autoregressively, and improvement

heuristics, which learn to refine tours. In this paper, we focus on the heatmap-guided MCTS paradigm, a hybrid of non-autoregressive edge probability generation and MCTS-based improvement. Methods in this paradigm can be grouped into two categories: supervised methods and label-free methods. Appendix A.1 surveys the broader classes of neural heuristics for TSP in more detail.

Supervised methods. These methods use a solver to generate labeled training datasets. On small scales, such as TSP-50, exact solvers can label large datasets efficiently, but they become intractable beyond a few thousand nodes, so supervised methods fall back on metaheuristics such as LKH-3 and train on tours that are themselves only near-optimal. Att-GCN (Fu et al., 2021) is an attention-based graph convolutional network (GCN) trained on exact solver solutions at 20 and 50 nodes and generalized to larger instances through graph sampling and merging. More advanced diffusion-based solvers such as DIFUSCO (Sun & Yang, 2024) improve heatmap quality through iterative denoising and can generate more informative heatmaps. Additionally, Fast-T2T (Li et al., 2024) attains competitive quality in fewer denoising steps via self-consistency. Both are trained at scales above 200 nodes, where optimal tours become prohibitively expensive to obtain, and therefore depend on LKH-3 labels. Pan et al. (2026) avoid neural networks entirely with GT-Prior, which estimates how often the optimal successor of a node is the node's k-th nearest neighbor and assigns that empirical probability to every k-th neighbor of a node. GT-Prior does not need learnable parameters and is robust across scales, but its heatmap generation still requires access to solver labels.

Label-free methods. These methods do not assume access to training solutions, avoiding reliance on a solver. DIMES (Qiu et al., 2022) trains with an RL meta-learning objective over an autoregressively factorized solution distribution, while UTSP (Min et al., 2023) trains a scattering attention GNN on a hand-crafted TSP-specific surrogate loss, but the objective is inflexible and does not scale beyond TSP-1000. Neither method competes with supervised methods at scale (Pan et al., 2026), and both are outperformed by SoftDist (Xia et al., 2024), a label-free method that tunes a single temperature parameter over the softmax of the distance matrix. SoftDist is the only method to couple heatmap generation with MCTS, tuning its temperature jointly with MCTS hyperparameters. No NCO method in this paradigm has a comparable mechanism, so all are trained on a surrogate loss decoupled from the tour length MCTS finds, a flaw that has drawn recent criticism of the paradigm (Xia et al., 2024).

3 Preliminaries

3.1 Problem Description and Heatmap Representation

Given a graph $\mathcal{G} = (V, E)$ with $V = \{v_1, \ldots, v_N\}$, the objective of the TSP is to find a route that visits each node exactly once with a minimal total distance $L(\pi) = \sum_{i \in V} D_{i,\pi(i)}$, where D_{ij} is the distance between nodes i and j and $\pi(i)$ is the successor of node i in the tour π. A tour may be represented as a permutation of the N nodes, but methods in the heatmap-guided MCTS paradigm instead represent it as an $N \times N$ matrix X, where $X_{ij} = 1$ if the edge from node i to j belongs to the tour, and $X_{ij} = 0$ otherwise. A *heatmap* Φ is the relaxation of this representation, assigning to each edge a score in $[0, 1]$, which reflects how likely it is that the edge belongs to an optimal tour. Because Φ specifies per-edge scores rather than a particular tour, it carries no feasibility guarantee, and MCTS is responsible for constructing a valid tour from it.

Rather than score all N^2 edges, we restrict the outgoing edges of each node i to its K nearest neighbors, denoted $\mathcal{N}_K(i)$, so the heatmap is an $N \times K$ matrix. This reduces the representation from $O(N^2)$ to $O(NK)$ and has minimal impact on the quality of the best attainable solution. For example, in the TSP-500 test set, 99.99% of the edges in optimal tours are within each node's 25 nearest neighbors, and 68% of the optimal edges are within the two nearest neighbors. This concentration on short edges is a strong regularity of Euclidean TSP, and Pan et al. (2026) show that it accounts for much of the heatmap-guided MCTS paradigm's performance: their parameter-free method, GT-Prior, attains the best gap of any heatmap-guided MCTS method on TSP-10000 instances. Thus, a heatmap is only useful insofar as it improves on this nearest-neighbor bias, which means identifying the 32% of optimal successors that distance alone cannot rank highly. Doing so requires a training signal that separates those edges from the many longer ones that do not belong in an optimal tour, and the only signal that has succeeded at scale is a set of pre-solved instances (Sun & Yang, 2024; Li et al., 2024).

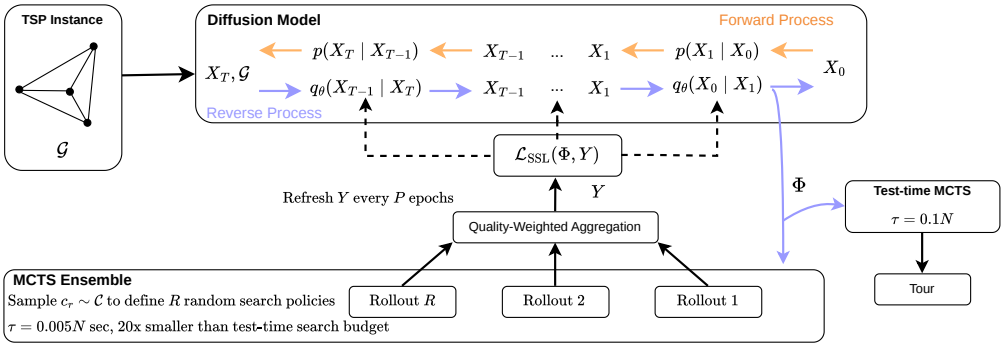


Figure 2: **Overview of MESCO.** A discrete diffusion model denoises a noisy solution X_T for a problem instance $\mathcal{G}$ over T steps, and the probabilities of the final step form the heatmap Φ. At test time, MCTS decodes a tour from Φ under a budget of $0.1N$ seconds per instance. During training, Φ instead seeds R short MCTS rollouts, each sampling a search configuration $c_r \sim \mathcal{C}$. A quality-weighted aggregation of chosen edges forms the target matrix Y, and $\mathcal{L}_{\text{SSL}}(\Phi, Y)$ propagates through every step of the denoising process. Targets are recomputed every P epochs, so improvements to the heatmap feed back into the rollouts that supervise it.

3.2 Monte Carlo Tree Search

MCTS is commonly used to decode TSP heatmaps, with the improvement-based implementation of Fu et al. (2021) used by most heatmap-guided MCTS methods. Search begins from an initial tour constructed by sampling edges with probability proportional to Φ_{ij}, followed by a 2-opt pass. It then samples k-opt moves according to a potential that balances heatmap-derived edge weights against an exploration bonus, accepting any move that shortens the tour and reinforcing the weights of the edges involved. Search restarts from a fresh initial tour whenever no improving move is found, and continues until a budget of τ seconds has elapsed. Its behavior is governed by six tunable parameters, of which the exploration coefficient α and the reinforcement coefficient β are the most consequential; Appendix A.6 gives the potential and update rules in full, along with the remaining parameters.

4 Method

We propose the MESCO framework, depicted in Figure 2. Given an instance $\mathcal{G}$, the diffusion model denoises a noisy solution X_T over T steps, and the marginals of the final step form the heatmap Φ, from which MCTS decodes a tour at test time under a search budget τ (in seconds). Most neural heatmap methods decouple search from training, so the network never learns from the tour discovered by MCTS, which is what determines performance at test time.

We close this gap with a self-supervision signal derived from MCTS itself. During training, we condition a noisy solution on a problem instance, denoise the solution, and extract the heatmap from the last denoising step. This heatmap then seeds R MCTS rollouts, each sampling its own search configuration and running at a fraction of the test-time budget. Individually these rollouts are weak and they disagree with one another, since a search policy that suits one instance may suit another poorly. That disagreement is what makes them useful in aggregate: edges on which the rollouts converge are those found under many different policies, while edges over which they split are those that the heatmap has yet to resolve. We aggregate these edges into a target matrix also in the form of a heatmap, recomputing this target every P epochs and applying supervision against the model's sampled heatmaps across the full denoising trajectory.

Section 4.1 describes how the MCTS ensembles are formed and turned into targets. Section 4.2 describes the discrete diffusion framework, which models a distribution over edge sets whose marginals form the heatmap, and provides the complete training objective. Since we supervised the heatmap directly during training, no discrete solution is required at any point.

Table 1: **Target quality against ensemble size at TSP-500.** Targets are built from R rollouts on a uniform prior at $\tau = 0.005N$ seconds each, scored against the optimal tour over 64 instances. Recall rises steadily with R while the rollout tours barely improve, so the aggregate recovers edges no individual rollout contains.

R	Wall (s) ↓	Gap ↓		Target recall ↑		
		Mean	Best	R@1	R@2	R@5
5	27.7	14.3%	8.9%	38.67	69.18	93.19
10	55.6	14.1%	8.1%	40.89	73.06	94.81
20	110.5	14.1%	7.7%	42.67	76.12	96.42
40	221.1	14.0%	7.1%	44.06	77.95	97.35

4.1 MCTS Ensembles

Supervised heatmap-guided MCTS methods depend on a solver such as Concorde or LKH-3 to label their training instances, and the cost of doing so grows with instance size until it dominates the training pipeline. Instead, we propose using MCTS ensembles to generate supervision, avoiding the need to solve TSP instances from scratch. While individual MCTS rollouts may find tours that are sub-optimal, the aggregate targets formed from MCTS ensembles can leverage search diversity to improve heatmap target quality, warranting their use as soft targets for heatmap supervision.

Forming MCTS ensembles. Given an instance $\mathcal{G}$ and its predicted heatmap Φ, we run R independent MCTS searches, each initialized from Φ and each allotted a budget of τ_{train}, which is a fraction of the budget τ used at test time. MCTS performance also depends strongly on its search parameters (Pan et al., 2026), and the best configuration varies from model to model. We therefore sample a fresh configuration c_r for each rollout from a space $\mathcal{C}$ over the MCTS parameters listed in Section 3.2, which produces diverse searches over the same instance at no additional cost.

Target construction. Each MCTS rollout r returns a tour π_r with length $\ell_r = L(\pi_r)$. From this tour π_r, we extract for each node i a one-hot indicator $v_i^{(r)} \in \{0,1\}^K$, which marks the candidate edge selected by π_r as the node's successor. Because shorter tours are more reliable evidence, rollouts are combined by a quality-weighted aggregation instead of a uniform average:

$$w_r = \frac{\exp\big(-(\ell_r - \ell_{\min})/(T_q\,\ell_{\min})\big)}{\sum_{j=1}^{R}\exp\big(-(\ell_j - \ell_{\min})/(T_q\,\ell_{\min})\big)}, \qquad \ell_{\min} = \min_{1\le j\le R}\ell_j, \tag{1}$$

so that a rollout's weight depends on its fractional gap to the best tour in the ensemble and the quality temperature T_q controls how sharply weight concentrates on the best rollouts. The target for node i is the weighted sum

$$Y_i = \sum_{r=1}^{R} w_r\, v_i^{(r)}, \tag{2}$$

where together, the vectors form a target matrix $Y \in [0,1]^{N\times K}$ with the same shape as Φ. Each row is a probability vector rather than a one-hot encoded indicator, which preserves coverage of optimal edges that are not assigned the highest probability: an edge chosen by every rollout receives probability close to 1, while an edge chosen by a minority receives probability in proportion to how often it was selected and how good the tours that selected it were.

Table 1 shows that ensembles recover optimal edges under a cheap budget even from a uniform heatmap. We measure target quality by R@k, the fraction of nodes whose optimal-tour successor appears among the k highest-weighted entries of that node's target row. Increasing R barely improves the mean rollout gap and improves the best-of-R gap only moderately, but it raises R@k substantially. Recall is what matters for supervision, since a target that covers more optimal edges gives the model more of them to learn from, and the divergence between the two shows that the ensemble is not simply finding better tours but different ones.

Refreshing MCTS targets. Targets are computed for every instance in the training set and cached before training begins, then recomputed every P epochs from the heatmaps the current model predicts. Each refresh therefore seeds its rollouts with a better prior than the last, closing a feedback

loop in which a better heatmap yields better-guided searches and those searches yield better targets. Algorithm 1 in Appendix A.2 gives the procedure in full. Two properties of Y carry over to Section 4.2: it is not a discrete solution, since its rows are marginals from which no tour is ever extracted, and it is not fixed, since it changes at every refresh.

4.2 Discrete Diffusion Framework for TSP Heatmaps

The targets of Section 4.1 are probabilities, so any model producing a probability per candidate edge could be trained with them. We use a discrete diffusion model because iterative refinement has proven the stronger architecture here: under identical supervision, the single-pass GCN of Fu et al. (2021) attains a 0.69% gap at TSP-500 while the diffusion model of Sun & Yang (2024) attains 0.33%. Sun & Yang (2024) add noise to tour solutions and learn to predict the added noise, which requires solving each training example from scratch before training; we never construct one, and instead supervise the marginals of the reverse process directly, as described below.

We model heatmap generation with a discrete diffusion process over the sparse edge matrix $X \in \{0,1\}^{N\times K}$ of Section 3.1. A forward process adds Bernoulli noise to a solution over T timesteps, mapping $P(X_0)$ to a uniform stationary distribution $q(X_T)$, and a learned reverse process denoises back to a sample, giving the joint distribution $q_\theta(X_{0:T}) = q(X_T)\prod_{t=1}^{T} q_\theta(X_{t-1} \mid X_t)$. With the solution flattened over the NK candidate edges, noise is applied independently to each entry, $P(X_t \mid X_{t-1}) = \prod_i p(X_{t,i} \mid X_{t-1})$, where

$$p(X_{t,i}) = \begin{cases} (1-\alpha_t)^{1-X_{t-1,i}} \cdot \alpha_t^{X_{t-1,i}} & \text{if } X_{t,i} = 0, \\ (1-\alpha_t)^{X_{t-1,i}} \cdot \alpha_t^{1-X_{t-1,i}} & \text{if } X_{t,i} = 1, \end{cases} \tag{3}$$

with $\alpha_t = \frac{1}{T-t+2}$. The heatmap is read as the marginals of the final denoising step, $\Phi_{ij} = q_\theta(X_{0,ij} = 1 \mid X_1)$, so that $\Phi \in [0,1]^{N\times K}$ assigns each candidate edge a probability of belonging to the tour.

Training discrete diffusion models without corrupted pairs. Supervised diffusion models are trained by adding noise to a ground-truth tour and learning to invert this noise (Sun & Yang, 2024; Li et al., 2024). Their objective is the standard discrete-diffusion variational bound, whose per-step terms depend on the forward posterior $q(X_{t-1} \mid X_t, X_0)$. That posterior is available in closed form only because X_0 is a known, feasible tour from Concorde or LKH-3. Our supervision provides no such X_0: the target Y is a set of marginals, and the rollout tours are discarded once Y is formed. Sanokowski et al. (2024) show that a discrete diffusion model can instead be trained by drawing samples from the reverse process itself, using an objective whose terms take the form

$$\mathcal{L}_{\text{diff}}(\theta; \mathcal{G}) = -\sum_{t=1}^{T} \mathbb{E}_{X_{T:t}}\big[S\big(q_\theta(X_{t-1} \mid X_t)\big)\big] \; - \; \sum_{t=1}^{T} \mathbb{E}_{X_{T:t-1}}\big[\log p(X_t \mid X_{t-1})\big], \tag{4}$$

where S denotes Shannon entropy. Both expectations are over trajectories of the reverse process, and we approximate them by drawing M trajectories from $q_\theta(X_{0:T} \mid \mathcal{G})$ for each instance. The first term prevents trajectory collapse during denoising, and the second term enforces consistency with the forward noise process of Eq. (4). Sanokowski et al. (2024) pair Eq. (4) with the expected energy of a Boltzmann distribution, minimizing a reverse KL divergence to a fixed target defined by a hand-specified energy function, which enables sampling of discrete solutions when ground-truth labels are unavailable. This framework could be applied to TSP, for which an energy function exists (Lucas, 2014), but doing so would substitute one hand-designed objective for another. Objectives of this kind are not competitive on TSP at scale: the unsupervised baseline UTSP crafts a similar TSP-specific surrogate loss, yet is outperformed by simple heuristic heatmaps (Xia et al., 2024; Pan et al., 2026). Supervision appears to close the gap, so we keep the supervision but replace only its source, supplying the task signal as a cross-entropy against soft targets built from MCTS ensembles.

Self-supervised training objective. The heatmap is compared against the target matrix of Eq. (4) by a soft-target binary cross-entropy. Let $p_{ik}^{(t,b)}$ denote the model's probability for candidate edge k of node i at denoising step t on instance $\mathcal{G}_b$, and $Y^{(b)}$ the target cached for that instance. The self-supervised loss on a single instance is

$$\mathcal{L}_{\text{SSL}}(\theta; \mathcal{G}_b) = \frac{1}{T\,N} \sum_{t=1}^{T} \sum_{i=1}^{N} \sum_{k=1}^{K} \Big[-\, Y_{ik}^{(b)} \log p_{ik}^{(t,b)} \; - \; \big(1 - Y_{ik}^{(b)}\big) \log\big(1 - p_{ik}^{(t,b)}\big) \Big], \tag{5}$$

which supervises the heatmap at every step of the denoising trajectory. Combining this loss with the diffusion terms keeps the reverse process a well-formed diffusion model while $\mathcal{L}_{\mathrm{SSL}}$ becomes the sole driver of solution quality, supervising the heatmap at every step of the denoising trajectory. Averaging over a batch of B instances drawn from a training distribution $\omega(\mathcal{G})$, the full objective is

$$\mathcal{L}(\theta) = \frac{1}{B}\sum_{b=1}^{B}\Big[\mathcal{T}\,\mathcal{L}_{\mathrm{diff}}(\theta;\mathcal{G}_b) + \lambda\,\mathcal{L}_{\mathrm{SSL}}(\theta;\mathcal{G}_b)\Big], \tag{6}$$

where λ controls the weight of self-supervision. The temperature $\mathcal{T}$ scales the diffusion terms and is annealed from $\mathcal{T}_{\max}$ to 0 over training. Early on, a high temperature keeps each denoising distribution high-entropy, preventing the model from committing to edge rankings while the targets are still uninformative, which is precisely when they are computed from a near-random heatmap. As $\mathcal{T}$ decreases, the regularizer fades, and the self-supervised term increasingly determines the heatmap.

5 Experiments

5.1 Experimental Settings

Datasets and baselines. We evaluate on the TSP-500, TSP-1000, and TSP-10000 test sets of Fu et al. (2021), which comprise 128 instances at the first two scales and 16 at the third, with node coordinates drawn uniformly from $[0, 1]^2$. We compute reference objectives $L(\pi^*)$ using Concorde (Applegate et al., 2006) for TSP-500 and TSP-1000 and using LKH 3 (Helsgaun, 2017) for TSP-10000. The optimality gap of a tour π is defined as $100 \cdot (L(\pi)/L(\pi^*) - 1)$. We compare against the exact solver Concorde and the heuristic solver LKH-3, both labeled **OR**, and the heatmap-guided MCTS methods Att-GCN (Fu et al., 2021), DIFUSCO (Sun & Yang, 2024) and Fast-T2T (Li et al., 2024), all supervised on solver-generated tours (**SL**); GT-Prior (Pan et al., 2026) and SoftDist (Xia et al., 2024), which are data-driven heuristics with no neural network (**DD**); DIMES (Qiu et al., 2022), trained by reinforcement learning (**RL**); and UTSP (Min et al., 2023), trained by unsupervised learning (**UL**). MESCO is self-supervised on MCTS-generated targets (**SSL**). We also include a non-informative Zero heatmap, which sets $\Phi_{ij} = 0$ for every edge. Baseline tour lengths and optimality gaps are reproduced from Pan et al. (2026), Table 1, whose per-method tuned MCTS configurations we also adopt unchanged rather than retuning. The heatmaps used for the rank-based analysis of Table 3 are the publicly released heatmaps of Xia et al. (2024), which cover every method except Fast-T2T. All runtimes are measured on our hardware with the exception of Fast-T2T, as noted in Appendix A.5.

MCTS decoding. All heatmap methods are decoded with the MCTS implementation of Fu et al. (2021), using the per-method configurations tuned by Pan et al. (2026). The search budget is $\tau = 50, 100, 1000$ seconds for TSP-500, TSP-1000, and TSP-10000, respectively. Each method receives the same MCTS budget, so differences in total runtime come from heatmap generation.

Training and hardware. We set $\tau_{\mathrm{train}} = 2.5, 5, 20$ seconds for TSP-500, TSP-1000, and TSP-10000, respectively. Model checkpoints are selected by the mean length of the initial tour that MCTS constructs from the heatmap on a held-out validation set, before any k-opt search. MESCO uses an AGNN as the denoising architecture, similar to DIFUSCO (Sun & Yang, 2024); Appendix A.3 describes the neural network architecture in more detail, and Appendix A.4 lists model hyperparameters and tuned MCTS hyperparameters for each scale. All experiments ran on a workstation with an Intel Core i9-10980XE CPU (18 cores, 36 threads), 128 GiB of RAM and one NVIDIA RTX 3090 GPU. Heatmap generation runs on the GPU; MCTS runs on the CPU and is parallelized across 36 threads.

5.2 Results on Large-Scale TSP

Table 2 reports optimality gaps and runtimes for MESCO and the heatmap-guided MCTS baselines. On TSP-500, MESCO is competitive with the state-of-the-art supervised heatmap methods, splitting DIFUSCO and Fast-T2T. MESCO outperforms DIFUSCO, which is trained on high-quality LKH-3 solutions, while generating heatmaps an order of magnitude faster on the same hardware. On TSP-1000, the gaps of supervised methods widen, while MESCO degrades only slightly: MESCO attains the best gap of any heatmap-guided MCTS method at this scale, at an inference cost comparable to label-free methods. The largest difference appears on TSP-10000, where MESCO attains a 1.10% gap, using only 3.60 minutes of heatmap generation, while the second-best method, GT-Prior, is at

Table 2: **Results on large-scale TSP.** Baseline tour lengths and gaps are from Pan et al. (2026), Table 1; all runtimes are measured on our hardware. Methods with two TIME terms report heatmap generation and MCTS decoding, respectively. † marks the reference objective at each scale, against which all gaps are measured. **Bold** marks the best gap at each scale and underline the second best. Test sets, reference objectives, search budgets and the TYPE legend are given in Section 5.1. *Fast-T2T heatmap times are quoted rather than measured; see Appendix A.5.

METHOD	TYPE	TSP-500			TSP-1000			TSP-10000		
		LENGTH ↓	GAP ↓	TIME ↓	LENGTH ↓	GAP ↓	TIME ↓	LENGTH ↓	GAP ↓	TIME ↓
CONCORDE	OR(EXACT)	16.55†	—	2.11m	23.12†	—	44.20m	—	—	—
LKH-3 (DEFAULT)	OR(HEURISTIC)	16.55	0.00%	3.74m	23.12	0.00%	7.50m	71.77†	—	97.78m
ZERO	MCTS	16.66	0.66%	0.00m+ 2.96m	23.39	1.16%	0.00m+ 5.93m	74.50	3.80%	0.00m+ 16.65m
SOFTDIST	DD+MCTS	16.62	0.43%	0.00m+ 2.96m	23.30	0.80%	0.00m+ 5.93m	73.89	2.95%	0.00m+ 16.65m
GT-PRIOR	DD+MCTS	16.63	0.50%	0.00m+ 2.96m	23.31	0.85%	0.00m+ 5.93m	73.31	2.14%	0.00m+ 16.65m
DIMES	RL+MCTS	16.66	0.43%	0.18m+ 2.96m	23.37	1.11%	0.29m+ 5.93m	73.97	3.06%	1.67m+ 16.65m
UTSP	UL+MCTS	16.69	0.90%	0.12m+ 2.96m	23.47	1.53%	0.19m+ 5.93m	—	—	—
ATT-GCN	SL+MCTS	16.66	0.69%	1.03m+ 2.96m	23.37	1.09%	1.94m+ 5.93m	73.95	3.03%	5.62m+ 16.65m
DIFUSCO	SL+MCTS	16.60	0.33%	2.62m+ 2.96m	23.24	0.53%	10.20m+ 5.93m	73.47	2.37%	13.51m+ 16.65m
FAST-T2T*	SL+MCTS	16.57	**0.12%**	0.50m+ 2.96m	23.27	0.65%	1.78m+ 5.93m	74.80	4.22%	7.73m+ 16.65m
MESCO (ours)	SSL+MCTS	16.59	0.22%	0.21m+ 2.96m	23.24	**0.51%**	0.40m+ 5.93m	72.56	**1.10%**	3.60m+ 16.65m

Table 3: **Rank-based heatmap analysis.** For each node, we rank its outgoing edges by heatmap score and report the recall of the optimal successor within the top-k (R@k), computed before MCTS decoding. Values are percentages. SoftDist and GT-Prior rank edges monotonically in distance, so their recall reflects the cumulative k-nearest-neighbor prior rather than anything learned, and is identical for both at every scale. Omissions are explained in Appendix A.5.

METHOD	TSP-500			TSP-1000			TSP-10000		
	R@1 ↑	R@2 ↑	R@5 ↑	R@1 ↑	R@2 ↑	R@5 ↑	R@1 ↑	R@2 ↑	R@5 ↑
SOFTDIST	44.02	69.56	94.25	43.76	69.36	94.46	44.59	70.46	94.86
GT-PRIOR	44.02	69.56	94.25	43.76	69.36	94.46	44.59	70.46	94.86
DIMES	41.34	62.38	84.76	40.26	60.34	82.84	41.95	63.78	87.55
UTSP	40.76	64.75	93.32	38.25	61.02	90.43	—	—	—
ATT-GCN	44.00	74.43	94.45	43.56	73.47	94.08	43.05	73.17	92.81
DIFUSCO	46.42	83.50	96.82	45.12	81.62	96.66	46.36	81.21	96.64
MESCO (ours)	**47.51**	**84.56**	**99.19**	**47.76**	**85.58**	**99.37**	**47.39**	**84.75**	**99.22**

2.14%. DIFUSCO and Fast-T2T continue to degrade, reaching gaps comparable to the data-driven priors. Existing supervised heatmap methods therefore lose the advantage that they have at smaller problem sizes once LKH-3 labels become more difficult to obtain at scale, while MESCO does not.

Table 3 examines heatmap quality directly. We sort each node's outgoing edges by predicted probability and report the recall of the optimal successor edge within the top-k, aggregated over all nodes and instances. Higher recall within a smaller candidate set indicates a better-focused search space for MCTS at test time. MESCO attains the highest R@k at every scale and every k, exceeding 99% at R@5 throughout. DIFUSCO's recall declines with scale, most visibly at R@2, while MESCO holds steady, which is reflected by the gaps in Table 2.

5.3 COST OF SUPERVISION

Table 4 compares the cost of generating supervision for the supervised diffusion models against MESCO. For each baseline, we calculate the CPU hours required to produce its training labels offline, using the published training-set sizes and our measured per-instance solver runtime, and compare this to the total MCTS runtime incurred by MESCO building targets online. For TSP-500 and TSP-1000, the baselines spend 1,600 and 1,902 CPU-hours on labels alone, before any training begins. For TSP-10000, the cost reaches roughly 8,500 CPU-hours for DIFUSCO's 6,400-instance

Table 4: **Training set sizes.** LABELS is the number of pre-solved instances each method requires and LABEL COST the CPU time to produce them with LKH-3 at our measured per-instance runtime. TARGET COST is the MCTS runtime MESCO spends building targets, $|\mathcal{D}| \times R \times \tau$ per refresh.

	TSP-500	TSP-1000	TSP-10000
Per-instance solver time			
Concorde (exact)	22.6 s	345 s	—
LKH-3 (heuristic)	45 s	107 s	79 min
Offline label generation			
DIFUSCO (Sun & Yang, 2024) labels	128,000	64,000	6,400
label cost (CPU hours)	1,600	1,902	8,452
Fast-T2T (Li et al., 2024) labels	128,000	64,000	—
label cost (CPU hours)	1,600	1,902	—
Online self-supervision (ours)			
Target cost (CPU hours)	333	278	44

Table 5: **Ablation of the MCTS ensemble and entropy regularization at TSP-500.** Each variant differs from MESCO in exactly one respect and shares the rest of the training settings.

VARIANT	R	REFRESHES	TARGET	ENTROPY REGULARIZER	TOUR LENGTH ↓	TOUR GAP ↓	HEATMAP R@1 ↑	HEATMAP R@5 ↑
Best-of-R	20	6	one-hot	✓	16.59	0.29%	46.70	98.80
$R = 5$	5	6	weighted	✓	16.59	0.29%	47.20	98.80
$R = 1$	1	6	one-hot	✓	16.60	0.31%	46.80	98.70
Single refresh	20	1	weighted	✓	16.63	0.52%	45.40	98.30
No entropy term	20	6	weighted		16.60	0.31%	47.08	98.79
MESCO	20	6	weighted	✓	**16.59**	**0.22%**	**47.51**	**99.19**

dataset, which MESCO reduces by $192\times$. The bottleneck also shapes what can be trained on: as problem size grows, supervised heatmap methods must shrink their training sets to keep labeling tractable. As a result, these models struggle to generalize at scale, especially on TSP-10000.

5.4 ABLATION ANALYSIS

Table 5 measures how the ensemble design affects performance on TSP-500 against four variants, each differing from MESCO in one respect. *Best-of-R* replaces the quality-weighted vote with a one-hot target from the shortest tour found; $R = 5$ and $R = 1$ reduce the ensemble size at a matched search budget; and *single refresh* computes targets once against the initial prior and never updates them. All four underperform MESCO, but not equally. Reducing the ensemble or collapsing it to a single tour costs 0.07 to 0.09 points of gap, and even $R = 5$ still beats $R = 1$, so aggregation contributes beyond what any one rollout provides. The refresh loop matters far more: freezing the targets more than doubles the gap to 0.52%, which is the only variant that DIFUSCO outperforms at this scale. Its heatmap is also the most diffuse of the five, which is what a model trained on targets from a near-random prior would be expected to produce. Removing the entropy regularizer costs 0.09 points, consistent with its role in maintaining exploration early in training.

6 CONCLUSION

We introduced MESCO, a self-supervised diffusion framework that learns to sample informative TSP heatmaps from MCTS ensembles. By placing MCTS inside the training loop, MESCO learns from the tours MCTS discovers rather than from a pre-solved dataset, aligning its training signal with the quantity that determines performance at test time and reducing the cost of supervision by up to $192\times$. The variance of MCTS across search configurations is easily leveraged: aggregating rollouts run under many different policies recovers optimal edges that no single rollout contains. The framework requires only a stochastic search that consumes a heatmap and returns decomposable solutions, so it should extend to other heatmap-guided pipelines; Appendix A.7 gives preliminary evidence on CVRP. That same variance also suggests a single tuned MCTS configuration leaves performance on the table at test time, and that learning a search policy per instance may be worthwhile.

AI USE STATEMENT

In this work, we used generative AI tools to assist with the presentation of the paper, including revising the manuscript, formatting tables, and improving readability, as well as for summarizing related work and writing and editing software code. All AI-assisted code edits were checked for correctness, and all summaries of prior work were verified against the original papers. The core method, experimental design, and results were not generated by AI tools. We take responsibility for the final content of this work, including any text, claims, or artifacts produced with AI assistance.

ETHICS STATEMENT

We believe this work does not violate the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

We provide our source code in the supplementary material, including scripts for training models, generating heatmaps from model checkpoints, and decoding those heatmaps with MCTS. A detailed README offers step-by-step instructions for setting up the environment and dependencies needed to reproduce the results in the paper. We additionally release the trained checkpoints for all three problem scales, together with the heatmaps they produce on the test sets, so that the reported optimality gaps can be reproduced by decoding the released heatmaps directly, without retraining. Checkpoints for the ablation variants of Section 5.4 are included on the same terms.

The experimental setup, including datasets, reference solvers, decoding budgets and hardware, is described in Section 5.1. Full training hyperparameters for each problem scale are given in Appendix A.4, alongside the MCTS configuration space sampled during target construction, the grid over which the decoder is tuned at evaluation, and the tuned configurations used to produce our reported results. The target construction procedure is specified in Section 4.1 and given as pseudocode in Appendix A.2. All training runs use a fixed random seed, reported with the configurations.

Test instances are taken from the established benchmarks of Fu et al. (2021); training instances are generated from the same uniform distribution and the generation code is included. Baseline tour lengths and optimality gaps are reproduced from Pan et al. (2026), whose tuned MCTS configurations we adopt unchanged. Unless stated otherwise, reported inference times for baselines are taken from our own hardware for fairness.

REFERENCES

David L. Applegate, Robert E. Bixby, Vašek Chvatál, and William J. Cook. *The Traveling Salesman Problem*. Princeton University Press, 2006. ISBN 978-0-691-12993-8. URL http://www.jstor.org/stable/j.ctt7s8xg.

Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning, 2017. URL https://openreview.net/forum?id=rJY3vK9eg.

Jieyi Bi, Yining Ma, Jiahai Wang, Zhiguang Cao, Jinbiao Chen, Yuan Sun, and Yeow Meng Chee. Learning generalizable models for vehicle routing problems via knowledge distillation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 31226–31238. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/ca70528fb11dc8086c6a623da9f3fee6-Paper-Conference.pdf.

Xinyun Chen and Yuandong Tian. Learning to Perform Local Rewriting for Combinatorial Optimization. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://papers.nips.cc/paper_files/paper/2019/hash/131f383b434fdf48079bff1e44e2d9a5-Abstract.html.

Jinho Choo, Yeong-Dae Kwon, Jihoon Kim, Jeongwoo Jae, André Hottung, Kevin Tierney, and Youngjune Gwon. Simulation-guided beam search for neural combinatorial optimization, 2022. URL https://arxiv.org/abs/2207.06190.

Paulo R. de O. da Costa, Jason Rhuggenaath, Yingqian Zhang, and Alp Akcay. Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning, 2020. URL https://arxiv.org/abs/2004.01608.

Zhang-Hua Fu, Kai-Bin Qiu, and Hongyuan Zha. Generalize a small pre-trained model to arbitrarily large tsp instances. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):7474–7482, May 2021. doi: 10.1609/aaai.v35i8.16916. URL https://ojs.aaai.org/index.php/AAAI/article/view/16916.

Keld Helsgaun. An Extension of the Lin-Kernighan-Helsgaun TSP Solver for Constrained Traveling Salesman and Vehicle Routing Problems. Technical report, Roskilde University, 2017.

André Hottung and Kevin Tierney. Neural Large Neighborhood Search for the Capacitated Vehicle Routing Problem, November 2020. URL http://arxiv.org/abs/1911.09539. arXiv:1911.09539 [cs].

Chaitanya K. Joshi, Thomas Laurent, and Xavier Bresson. An Efficient Graph Convolutional Network Technique for the Travelling Salesman Problem, October 2019. URL http://arxiv.org/abs/1906.01227. arXiv:1906.01227 [cs].

Chaitanya K. Joshi, Quentin Cappart, Louis-Martin Rousseau, and Thomas Laurent. Learning TSP requires rethinking generalization. *LIPIcs, Volume 210, CP 2021*, 210:33:1–33:21, 2021. doi: 10.4230/LIPICS.CP.2021.33. URL https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.CP.2021.33.

Minsu Kim, Junyoung Park, and Jinkyoo Park. Sym-nco: Leveraging symmetricity for neural combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 35, pp. 1936–1949. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/0cddb777d3441326544e21b67f41bdc8-Paper-Conference.pdf.

Minsu Kim, Sanghyeok Choi, Hyeonah Kim, Jiwoo Son, Jinkyoo Park, and Yoshua Bengio. Ant colony sampling with gflownets for combinatorial optimization. In Yingzhen Li, Stephan Mandt, Shipra Agrawal, and Emtiyaz Khan (eds.), *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pp. 469–477. PMLR, 03–05 May 2025. URL https://proceedings.mlr.press/v258/kim25a.html.

Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ByxBFsRqYm.

Wouter Kool, Herke van Hoof, Joaquim Gromicho, and Max Welling. Deep Policy Dynamic Programming for Vehicle Routing Problems, December 2021. URL http://arxiv.org/abs/2102.11756. arXiv:2102.11756 [cs].

Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. POMO: Policy Optimization with Multiple Optima for Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21188–21198. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/f231f2107df69eab0a3862d50018a9b2-Abstract.html.

Gilbert Laporte. The traveling salesman problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(2):231–247, 1992. ISSN 0377-2217. doi: https://doi.org/10.1016/0377-2217(92)90138-Y. URL https://www.sciencedirect.com/science/article/pii/037722179290138Y.

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*, 2024.

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.

Hao Lu, Xingwen Zhang, and Shuang Yang. A learning-based iterative method for solving vehicle routing problems. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJe1334YDH.

Andrew Lucas. Ising formulations of many np problems. *Frontiers in Physics*, 2, 2014. ISSN 2296-424X. doi: 10.3389/fphy.2014.00005. URL http://dx.doi.org/10.3389/fphy.2014.00005.

Yining Ma, Jingwen Li, Zhiguang Cao, Wen Song, Le Zhang, Zhenghua Chen, and Jing Tang. Learning to Iteratively Solve Routing Problems with Dual-Aspect Collaborative Transformer. In *Advances in Neural Information Processing Systems*, volume 34, pp. 11096–11107. Curran Associates, Inc., 2021. URL https://papers.nips.cc/paper_files/paper/2021/hash/5c53292c032b6cb8510041c54274e65f-Abstract.html.

Yimeng Min, Yiwei Bai, and Carla P Gomes. Unsupervised learning for solving the travelling salesman problem. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=lAEc7aIW20.

Alex Nowak, Soledad Villar, Afonso S. Bandeira, and Joan Bruna. Revised note on learning algorithms for quadratic assignment with graph neural networks, 2018. URL https://arxiv.org/abs/1706.07450.

Xuanhao Pan, Yan Jin, Yuandong Ding, Mingxiao Feng, Li Zhao, Lei Song, and Jiang Bian. H-TSP: Hierarchically Solving the Large-Scale Traveling Salesman Problem. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(8):9345–9353, June 2023. ISSN 2374-3468. doi: 10.1609/aaai.v37i8.26120. URL https://ojs.aaai.org/index.php/AAAI/article/view/26120. Number: 8.

Xuanhao Pan, Chenguang Wang, Chaolong Ying, Ye XUE, and Tianshu Yu. Beyond the heatmap: A rigorous evaluation of component impact in MCTS-based TSP solvers. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=H6PLJnnK6e.

Petrică C. Pop, Ovidiu Cosma, Cosmin Sabo, and Corina Pop Sitar. A comprehensive survey on the generalized traveling salesman problem. *European Journal of Operational Research*, 314(3):819–835, 2024. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2023.07.022. URL https://www.sciencedirect.com/science/article/pii/S0377221723005581.

Ruizhong Qiu, Zhiqing Sun, and Yiming Yang. DIMES: A differentiable meta solver for combinatorial optimization problems. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=9u05zr0nhx.

Sebastian Sanokowski, Sepp Hochreiter, and Sebastian Lehner. A diffusion model framework for unsupervised neural combinatorial optimization. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=AFfXlKFHXJ.

Zhiqing Sun and Yiming Yang. Difusco: graph-based diffusion solvers for combinatorial optimization. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2024. Curran Associates Inc.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer Networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://papers.nips.cc/paper_files/paper/2015/hash/29921001f2f04bd3baee84a12e98098f-Abstract.html.

Xuan Wu, Di Wang, Lijie Wen, Yubin Xiao, Chunguo Wu, Yuesong Wu, Chaoyu Yu, Douglas L. Maskell, and You Zhou. Neural combinatorial optimization algorithms for solving vehicle routing problems: A comprehensive survey with perspectives, 2025. URL https://arxiv.org/abs/2406.00415.

Yaoxin Wu, Wen Song, Zhiguang Cao, Jie Zhang, and Andrew Lim. Learning Improvement Heuristics for Solving Routing Problems, May 2020. URL http://arxiv.org/abs/1912.05784. arXiv:1912.05784 [cs].

Yifan Xia, Xianliang Yang, Zichuan Liu, Zhihao Liu, Lei Song, and Jiang Bian. Position: rethinking post-hoc search-based neural approaches for solving large-scale traveling salesman problems. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Haoran Ye, Jiarui Wang, Zhiguang Cao, Helan Liang, and Yong Li. DeepACO: Neural-enhanced ant systems for combinatorial optimization. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=cd5D1DD923.

Haoran Ye, Jiarui Wang, Helan Liang, Zhiguang Cao, Yong Li, and Fanzhang Li. Glop: learning global partition and local construction for solving large-scale routing problems in real-time. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press, 2024. ISBN 978-1-57735-887-9. doi: 10.1609/aaai.v38i18.30009. URL https://doi.org/10.1609/aaai.v38i18.30009.

A Appendix

A.1 Extended Related Work

Neural approaches to TSP are conventionally grouped by how a solution is produced. *Construction methods* build a tour incrementally, adding one node at a time until the route is complete. *Improvement methods* begin from a complete tour and apply local operators repeatedly until a computational budget is exhausted. The heatmap-guided MCTS paradigm studied in this paper draws on both, using a network to score every edge in a single pass and an improvement-based search to construct and refine a tour from those scores.

Construction methods. Vinyals et al. (2015) introduced the Pointer Network, which decodes a tour autoregressively with a recurrent architecture trained on pre-computed solutions. Bello et al. (2017) replaced that supervision with reinforcement learning, using the negative tour length as a reward. Transformer architectures (Vaswani et al., 2017) subsequently displaced the recurrent decoder (Kool et al., 2019; Kwon et al., 2020; Kim et al., 2022), with further work on training procedures (Bi et al., 2022) and decoding strategies (Choo et al., 2022). These methods are effective up to roughly 100 nodes but scale poorly beyond it (Joshi et al., 2021; Fu et al., 2021), since the action space grows linearly in N while self-attention costs $O(N^2)$ per step. One response is to decompose the problem: Pan et al. (2023) solve large instances hierarchically as a set of smaller open-loop sub-problems, which reduces training cost at a substantial price in solution quality, reporting gaps of 6.62% at TSP-1000 and 7.32% at TSP-10000.

Improvement methods. Improvement-based solvers learn to guide classical local operators rather than to construct tours. Representative operators include learned local search (Chen & Tian, 2019; Lu et al., 2020), ruin-and-repair (Hottung & Tierney, 2020), and 2-opt (de O. da Costa et al., 2020; Wu et al., 2020; Ma et al., 2021). Their principal difficulty at scale is sparse reward: on large graphs, most local moves leave the objective essentially unchanged, so the learning signal is weak (Joshi et al., 2021). Ye et al. (2024) address this by decomposing instances and improving the sub-problems, again trading solution quality for tractability.

Heatmap-guided search. The paradigm we work in combines non-autoregressive heatmap generation with an improvement-based search. A network scores every edge in a single non-autoregressive pass (Nowak et al., 2018; Joshi et al., 2019; Kool et al., 2021), and a search consumes those scores as a prior. The scoring model may be trained by supervised learning (Fu et al., 2021; Sun & Yang, 2024; Li et al., 2024), by reinforcement learning (Qiu et al., 2022), or by an unsupervised surrogate loss (Min et al., 2023). The decoder varies as well: MCTS is standard for TSP at scale (Fu et al., 2021), but beam search (Joshi et al., 2019), dynamic programming (Kool et al., 2021), ant colony optimization (Ye et al., 2023; Kim et al., 2025), and guided tree search (Li et al., 2018) have all been paired with learned priors, the last of these for maximum independent set rather than routing. All of these approaches ultimately supply a learned prior over solution components and the search, not the network, produces the solution.

Search in the training loop. Search has also been used inside the training loop, though to different ends than ours. DeepACO (Ye et al., 2023) and GFACS (Kim et al., 2025) apply local search to sampled solutions and train on the refined trajectories, with GFACS additionally reshaping the energy of each sample using the energy of its locally-searched refinement. In both cases the search improves individual training samples, and the learning signal remains the objective value of those samples. We instead aggregate across many independent searches on the same instance and distill their agreement into a per-node target, so that the signal is the collective behavior of an ensemble rather than the quality of any one solution.

A.2 MCTS Target Refresh Algorithm

Algorithm 1 gives the MCTS target refresh procedure in full. Each rollout casts one hard successor vote per node, and the target is their quality-weighted average, a distribution over the K candidates rather than a single tour. The refresh is invoked every P epochs; between refreshes the targets are fixed and the self-supervised term of Eq. (6) reduces to a binary cross-entropy against them,

alongside the diffusion loss of Eq. (6). The forward pass at line 2 is taken without gradients, and the symmetrization at line 3 reflects that MCTS operates on an undirected edge weighting.

Algorithm 1: MCTS target refresh

Input: dataset $\mathcal{D}$, parameters θ, rollouts R, budget τ, configuration space $\mathcal{C}$, quality temperature T_q
Output: target matrix $Y^{(b)} \in [0,1]^{N\times K}$ for every $\mathcal{G}_b \in \mathcal{D}$

1 **foreach** $\mathcal{G}_b \in \mathcal{D}$ **do**
2 $\quad \Phi \leftarrow f_\theta(\mathcal{G}_b)$; $\quad \Phi^{\mathrm{sym}} \leftarrow \frac{1}{2}(\Phi + \Phi^\top)$
3 $\quad$ **for** $r = 1, \dots, R$ **do**
4 $\quad\quad c_r \sim \mathrm{Uniform}(\mathcal{C})$; $\quad \pi_r \leftarrow \textsc{MctsSolve}(\Phi^{\mathrm{sym}}, \tau; c_r)$; $\quad \ell_r \leftarrow L(\pi_r)$
5 $\quad\quad v_{ik}^{(r)} \leftarrow \mathbf{1}\big[\mathrm{nn}_k(i) = \pi_r(i)\big]$ `// hard vote`
6 $\quad$ **end**
7 $\quad \ell_{\min} \leftarrow \min_{1\le j\le R} \ell_j$
8 $\quad w_r \leftarrow \mathrm{softmax}_r\big(-(\ell_r - \ell_{\min})/(T_q\,\ell_{\min})\big)$
9 $\quad Y^{(b)} \leftarrow \sum_{r=1}^{R} w_r\, v^{(r)}$
10 **end**
11 **return** $\{Y^{(b)}\}$

A.3 Denoising Architecture

Following DIFUSCO (Sun & Yang, 2024) and Fast-T2T (Li et al., 2024), we adopt an anisotropic graph neural network as the denoising model. Each layer updates edge embeddings from the embeddings of their incident nodes together with sinusoidal features of the current timestep t, and node embeddings are then updated through gated message passing that makes aggregation edge-aware. The inputs at timestep t are sinusoidal features of the two-dimensional node coordinates, the discrete edge matrix X_{t+1} from the previous denoising step, and the sinusoidal embedding of t. Layer counts, hidden dimensions, and the remaining architectural settings are listed for each problem scale in Table 6.

A.4 Hyperparameters

Table 6 lists the settings used at each problem scale. Values common to all scales are given once; the remainder vary with N and are reported per scale. Refresh cost is the dominant term at large N, so the number of instances and rollouts is reduced as N grows while the total number of supervised rows, $|\mathcal{D}| \times N$, is held roughly constant.

Table 6: Hyperparameters used at each problem scale.

Symbol	Meaning	TSP-500	TSP-1000	TSP-10000
Model and training				
K	kNN candidate set size	25	25	25
T	denoising steps	3	3	3
M	reverse-process samples per graph	2	2	1
–	AGNN layers	8	8	8
–	hidden dimension	64	64	64
B	batch size (graphs)	8	4	1
η	learning rate	2×10^{-3}	1×10^{-3}	1×10^{-3}
$\mathcal{T}_{\max}$	initial annealing temperature	0.1	0.001	0.001
–	annealing epochs	250	150	100
λ	self-supervision coefficient	200	200	200
MCTS target construction				
$\lvert\mathcal{D}\rvert$	training graphs	4000	2000	200
R	rollouts per graph	20	20	10
τ	MCTS budget per rollout (s)	2.5	5	20
P	refresh interval (epochs)	40	30	25
T_q	quality-weighting temperature	0.05	0.05	0.05

Search configuration spaces. Table 7 lists the MCTS configuration space used at evaluation, following Pan et al. (2026), alongside the space $\mathcal{C}$ sampled at each rollout during target construction. The training space is identical across problem scales and deliberately differs from the evaluation grid in two respects. We fix `Use_Heatmap` = `True` rather than sampling it, since a rollout that ignores the heatmap builds its candidate sets from nearest-neighbor proximity alone and so reports nothing about the model's learned rankings, which is what the targets are meant to capture. We also exclude $\alpha = 0$, which corresponds to pure exploitation of the current prior and therefore cannot surface edges that the prior does not already rank highly; excluding it measurably improves the recall of distant optimal edges in the resulting targets. Both parameters remain part of the evaluation grid, where the tuner is free to select them.

Table 7: MCTS configuration spaces. The evaluation grid is tuned per method and per scale by grid search (Pan et al., 2026); the training space is sampled uniformly and independently for each of the R rollouts used to construct a target.

PARAMETER	EVALUATION GRID	TRAINING SPACE $\mathcal{C}$
`Alpha`	$\{0, 1, 2\}$	$\{1, 2\}$
`Beta`	$\{10, 100, 150\}$	$\{10, 100, 150\}$
`Max_Depth`	$\{10, 50, 100, 200\}$	$\{10, 50, 100, 200\}$
`Max_Candidate_Num`	$\{5, 20, 50, 1000\}$	$\{5, 20, 50, 1000\}$
`Param_H`	$\{2, 5, 10\}$	$\{2, 5, 10\}$
`Use_Heatmap`	$\{$`True`, `False`$\}$	$\{$`True`$\}$
τ (s)	50 (TSP-500), 100 (TSP-1000), 1000 (TSP-10000)	2.5 (TSP-500), 5 (TSP-1000), 20 (TSP-10000)

Tuned search configuration for MESCO. We tune the decoding hyperparameters of MESCO with the same grid search and the same tuning set used for the baselines, so that no method receives an advantage from its search configuration. The selected settings are given in Table 8.

Table 8: MCTS configuration selected for MESCO at each scale, obtained by grid search over the evaluation grid of Table 7. Baseline configurations are those published by Pan et al. (2026).

SCALE	`Alpha`	`Beta`	`Max_Depth`	`Max_Candidate_Num`	`Param_H`	`Use_Heatmap`
TSP-500	0	150	200	5	2	`True`
TSP-1000	0	10	200	50	5	`True`
TSP-10000	2	10	200	20	2	`True`

A.5 BASELINE DETAILS

Baseline tour lengths and optimality gaps are reproduced from Pan et al. (2026), Table 1, whose per-method tuned MCTS configurations we adopt unchanged rather than retuning. Concorde, LKH-3 and every baseline heatmap generator were rerun on our own hardware, with the exception noted below. For the rank-based analysis of Table 3 we use the publicly released heatmaps of Xia et al. (2024).

Fast-T2T. Its heatmap generation times in Table 2 are quoted from Pan et al. (2026) rather than measured. The released checkpoints correspond to several model variants and the paper does not specify which produced the cited results, so the matching configuration cannot be reproduced. Those timings were obtained on different hardware and are not directly comparable to the remaining rows. For the same reason, Xia et al. (2024) release no heatmaps for Fast-T2T and it is omitted from Table 3.

UTSP. No model checkpoint is released at TSP-10000, so UTSP appears only at TSP-500 and TSP-1000.

A.6 MCTS DETAILS

We use the improvement-based MCTS of Fu et al. (2021) without modification. Given a heatmap Φ, an edge weight matrix is initialized as $W_{ij} = 100 \cdot \Phi_{ij}$ and an access frequency matrix Q is initialized to zero. At each step the search samples a set of k-opt moves according to the potential

$$Z_{ij} = \frac{W_{ij}}{\Omega_i} + \alpha \sqrt{\frac{\ln(C+1)}{Q_{ij}+1}}, \tag{7}$$

where α controls exploration, $\Omega_i = \sum_{j \neq i} W_{ij}$ normalizes the weights of node i, and C is a move counter. When a sampled move yields a shorter tour π' it is accepted and the weights of the participating edges are reinforced,

$$W_{ij} \leftarrow W_{ij} + \beta \left(\exp\left(\frac{L(\pi) - L(\pi')}{L(\pi)} \right) - 1 \right), \tag{8}$$

where β controls the update magnitude. If no improving move is found the search restarts from a new initial tour biased by the updated weights, and continues until the time limit τ is reached.

The search exposes six tunable parameters: the exploration coefficient α, the update coefficient β, the maximum k for k-opt moves (`Max_Depth`), the candidate edge set size per node (`Max_Candidate_Num`), the number of simulations per move (`Param_H`), and whether the heatmap is used to construct the initial candidate sets (`Use_Heatmap`). Table 7 lists the grid over which these are tuned at evaluation alongside the space we sample from during target construction.

A.7 GENERALIZATION TO LARGE-SCALE CVRP

The MESCO framework is not specific to TSP or MCTS. It requires only a sparse candidate set over which a solution decomposes, and a stochastic search that consumes a heatmap and returns complete solutions. To test this, we apply it to the capacitated vehicle routing problem (CVRP), replacing MCTS with ant colony optimization (ACO).

Adapting MESCO to CVRP. The diffusion model and training objective are left unchanged. Three components differ, all through configuration rather than a separate model. Node features grow from the two coordinates to four, adding normalized demand and a depot indicator, with the positional embedding applied to the coordinates and a separate dense layer to the remaining two. The sparse candidate set is defined over customers only: each customer has K outgoing edges, of which one slot is reserved for a return to the depot and the remainder are its $K - 1$ nearest customers, so a solution remains a binary matrix whose rows select a route successor. Target construction calls ACO in place of MCTS. Everything else, including the Bernoulli forward process, the three reverse steps, the AGNN backbone, and the soft-target cross-entropy of Eq. 5, is shared with the TSP model.

Ensembles from ACO. We build ensembles with the same ACO and local search implementation used at evaluation, following Ye et al. (2023) and Kim et al. (2025). Each rollout samples its own configuration over the pheromone weight, heuristic weight, evaporation rate and ant count, and runs at a budget far below the one used at decoding. The tours returned are aggregated exactly as in Eq. 2, weighting each rollout by the total distance of its solution.

Setup and results. We train at CVRP-500 from scratch and fine-tune to CVRP-1000, following the same procedure used for TSP, and decode with the ACO and neural-guided local search configuration of Kim et al. (2025) at $K = 100$ ants and $T = 10$ iterations so that our numbers are directly comparable to theirs. Experiments run on a machine with two AMD EPYC 7763 processors (128 cores, 256 threads), 503 GiB of RAM and one NVIDIA A100 with 80 GiB. Table 9 reports the results. MESCO attains gaps of 1.73% and 1.91% against PyVRP, ahead of GFACS at 1.95% and 2.11%, under an identical decoding budget at both scales. We report this as evidence that the method transfers rather than as a thorough evaluation of CVRP, for which a broader set of scales and baselines would be required.

Table 9: **Generalization to large-scale CVRP.** Objectives and gaps are reproduced from Kim et al. (2025), Table 2, with PyVRP as the reference solver; the PyVRP and LKH-3 runtimes are also theirs. All ACO methods, including ours, share the same ACO and local search implementation under an identical budget, and their runtimes are measured on our hardware as the average time to solve a single instance.

METHOD	CVRP-500			CVRP-1000		
	OBJ. ↓	GAP ↓	TIME ↓	OBJ. ↓	GAP ↓	TIME ↓
PyVRP	62.96	—	21.0m	119.29	—	1.4h
LKH-3	63.07	0.18%	18.6m	119.52	0.20%	2.1h
ACO	64.62	2.63%	75s	122.82	2.96%	4.06m
DeepACO	64.49	2.43%	75s	122.15	2.40%	4.06m
GFACS	64.19	<u>1.95%</u>	75s	121.80	<u>2.11%</u>	4.06m
MESCO	64.05	**1.73%**	75s	**121.57**	**1.91%**	4.06m