EDISCO: EQUIVARIANT CONTINUOUS-TIME CATEGORICAL DIFFUSION FOR GEOMETRIC COMBINATORIAL OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Geometric combinatorial optimization problems, such as the Traveling Salesman Problem (TSP), possess inherent symmetries under rotations, translations, and reflections in Euclidean space. These transformations are denoted as E(2). However, existing neural network-based approaches, including recent diffusion-based solvers, fail to exploit these geometric features. This paper presents EDISCO, to the best of our knowledge, the first diffusion-based framework combining E(2)-equivariant graph neural networks with continuous-time categorical diffusion models for solving geometric combinatorial problems. This approach introduces an equivariant score network that respects geometric transformations while operating on discrete edge variables, together with a continuous-time categorical diffusion process that maintains E(2) symmetries throughout the forward and reverse processes. By incorporating geometric awareness directly into the diffusion process, EDISCO achieves notable improvements over the baseline. EDISCO reduces the state-of-the-art TSP optimality gaps on TSP-500 from 0.12% to 0.08%, TSP-1000 from 0.30% to 0.22%, and TSP-10000 from 2.68% to 1.20%. EDISCO demonstrates strong generalizability across problem sizes and also shows remarkable efficiency, requiring only 33% to 50% of the training data compared to competing diffusion methods across all problem scales.

1 INTRODUCTION

With diverse applications in logistics, circuit design, and resource allocation, geometric combinatorial optimization problems (GCOPs), such as the Traveling Salesman Problem (TSP), remain a fundamental challenge in combinatorial optimization. Despite decades of research on exact and heuristic solvers (Applegate et al., 2006; Helsgaun, 2017), the development of learning-based approaches has recently become the focus because of their potential for rapid inference and generalization across various problem instances (Kool et al., 2019; Joshi et al., 2022; Fu et al., 2021). Recent breakthroughs in diffusion models have opened new directions for solving GCOPs (Graikos et al., 2022; Sun & Yang, 2023; Zhao et al., 2024). DIFUSCO (Sun & Yang, 2023) demonstrated graph-based diffusion for TSP, while DISCO (Zhao et al., 2024) introduced residue-constrained generation and analytical denoising to achieve up to 5.28× speedup over previous diffusion approaches.

However, an important observation is that TSP and similar GCOPs possess natural symmetries. The solutions to these problems remain invariant under transformations in 2D Euclidean space, including translations, rotations, and reflections (Ouyang et al., 2021; Bronstein et al., 2021). Such transformations are denoted as E(2). However, most existing neural network-based approaches fail to capture the geometric structure of TSP. These methods require massive amounts of training data and depend on high-quality optimal or near-optimal solutions for supervision, which are computationally expensive to obtain for large problems (Kool et al., 2019; Kwon et al., 2020; Joshi et al., 2022). Furthermore, these models face significant memory and computational constraints. Even moderate-scale problems cause memory overflow and require extensive training times (Bresson & Laurent, 2021; Xin et al., 2021; Fu et al., 2021). This inefficiency originates from their need to learn geometric invariances from scratch. Non-equivariant models attempt to address this issue with data augmentation, but this only shifts the problem. They require more training samples and still cannot ensure exact equivariance, especially on out-of-distribution data (Nordenfors et al., 2023; Esteves

et al., 2018). In contrast, models that explicitly incorporate geometric structure through equivariant architectures achieve better performance with less training data and smaller model sizes (Brehmer et al., 2024; Satorras et al., 2021; Batzner et al., 2022).

In addition to the geometric considerations, the choice of diffusion formulation presents another crucial design decision. While discrete diffusion models have shown promise for combinatorial problems (Sun & Yang, 2023; Austin et al., 2021), existing approaches employ discrete-time formulations with certain limitations. Although methods like DDIM (Song et al., 2021a) allow variable step counts at inference, they still rely on fixed discretization schemes that may accumulate approximation errors (Zhang & Chen, 2022b; Lu et al., 2022a; Ren et al., 2025). The discrete-time framework also limits access to adaptive numerical solvers that could dynamically adjust computational effort based on local dynamics (Ren et al., 2025; Zhang et al., 2024). Continuous-time formulations fundamentally address these limitations by treating the diffusion as a continuous process governed by stochastic differential equations (SDEs). This enables the use of numerical solvers with adaptive step sizes and higher-order integration methods that can achieve better accuracy with fewer number of function evaluations (NFEs) (Song et al., 2021a; Sun et al., 2023a).

In this work, we propose EDISCO, which leverages an Equivariant Graph Neural Network (EGNN) architecture to respect the geometric symmetries inherent in TSP instances. We also formulate the edge selection process as a continuous-time diffusion over discrete variables, enabling the derivation of analytical expressions for both the forward corruption process and the reverse denoising process, resulting in accelerated inference.

The novel contributions are as follows:

1. We introduce, to the best of our knowledge, the first continuous-time discrete diffusion model with built-in geometric equivariance for combinatorial optimization. Preserving problem symmetries improves both sample efficiency and solution quality.
2. We develop efficient training and sampling algorithms that leverage the analytical tractability of Continuous-time Markov Chains (CTMCs), compatible with higher-order accelerated solvers. These solvers achieve 2-3× speedups with better quality or up to 25× speedups for real-time applications compared to discrete-time methods.
3. The state-of-the-art performance is exceeded on TSP benchmarks (50-10000 cities). The optimality gap is reduced from 0.12% to 0.08% for TSP-500, from 0.30% to 0.22% for TSP-1000, and from 2.68% to 1.20% for TSP-10000. It only requires 33% to 50% of the training data compared to competing diffusion methods across all problem scales.
4. EDISCO is extended to solve real-world TSP problems and the Capacitated Vehicle Routing Problem (CVRP), outperforming SOTA approaches. These results validate the generalizability of EDISCO, indicating that it can be effectively applied to other GCOPs.

The remainder of this paper is organized as follows. Section 2 reviews related work on neural combinatorial optimization and diffusion models. Section 3 presents the continuous-time diffusion framework and the EGNN architecture. Section 4 presents comprehensive experimental results. Section 5 concludes with discussions and future directions.

2 Related Work

2.1 Neural Network-Based TSP Solvers

Neural network-based approaches for TSP have recently become mainstream due to their potential for rapid inference, generalization across problem instances, and ability to learn from data without hand-crafted heuristics (Kool et al., 2019; Joshi et al., 2022; Fu et al., 2021). These approaches can be divided into autoregressive and non-autoregressive models. Autoregressive models (Kool et al., 2019; Kwon et al., 2020) construct solutions sequentially but require extensive training data and struggle with generalization (Joshi et al., 2022). Non-autoregressive approaches generate complete solutions simultaneously, evolving from limited heatmap representations (Joshi et al., 2019) to expressive diffusion models (Sun & Yang, 2023; Li et al., 2023; Yoon et al., 2024; Zhao et al., 2024).

Among diffusion approaches, DIFUSCO (Sun & Yang, 2023) pioneered graph-based diffusion for TSP, DISCO (Zhao et al., 2024) achieved 5.28× speedup through residue-constrained generation, and T2T (Li et al., 2023) improved quality via gradient-based search. CADO (Yoon et al., 2024) combines enhanced DIFUSCO by combining RL fine-tuning, but requires high-quality supervised data and expensive RL fine-tuning. The key insight is that all existing diffusion TSP solvers use discrete-time diffusion formulations and ignore TSP's geometric structure.

2.2 Geometric Deep Learning and Equivariance

Geometric deep learning leverages symmetries to improve efficiency and generalization (Bronstein et al., 2021). Equivariant neural networks ensure outputs transform consistently with symmetric inputs, reducing sample complexity (Cohen & Welling, 2016). E(2)-equivariant models significantly improve TSP generalization (Ouyang et al., 2021), and geometric GNNs outperform standard architectures for TSP tasks (Song et al., 2025). Theory confirms that equivariant models reduce training data requirements (Brehmer et al., 2024). Despite this evidence, most neural network-based TSP solvers lack geometric awareness. While Sym-NCO (Kim et al., 2022) uses regularizer-based symmetry learning, it doesn't achieve exact equivariance. These gaps motivate our approach as the first to combine exact E(2)-equivariance with diffusion.

2.3 Continuous-Time Diffusion Formulations

Continuous-time formulations resolve fundamental limitations of discrete-time diffusion. CTMCs for discrete diffusion denoising (Campbell et al., 2022) enable analytical transition probabilities and flexible inference without retraining. Score-based continuous-time discrete diffusion (Sun et al., 2023b) provides better convergence properties and allows the use of higher-order integration methods that significantly reduce the number of neural network evaluations (Song et al., 2021b). Though DiffUCO (Sanokowski et al., 2024) applies continuous-time to unsupervised combinatorial optimization, it lacks geometric considerations and does not show results on large-scale geometric problems. DISCO (Zhao et al., 2024) also achieves fast inference (1-2 steps) by replacing the entire numerical integration process with an analytically solvable form through decoupled diffusion models (DDMs). This analytical solution completely bypasses the need for numerical ODE solvers but requires problem-specific residue constraints and sacrifices flexibility in the diffusion process.

3 Method

3.1 Problem Formulation

GCOPs in Euclidean space are defined on a set of n nodes $\mathcal{V}$ with coordinates $\{\mathbf{c}_i\}_{i=1}^n$, $\mathbf{c}_i \in \mathbb{R}^d$. The objective is to select a subset of edges or configurations, represented by a decision matrix $\mathbf{X} \in \{0,1\}^{n\times n}$, that minimizes a distance-based cost function while satisfying problem-specific constraints. The objective is:

$$\mathbf{X}^* = \arg\min_{\mathbf{X}} f(\mathbf{X}, \{\mathbf{c}_i\}_{i=1}^n) \quad \text{s.t. } \mathbf{X} \in \mathcal{C} \tag{1}$$

where f is a distance-based cost function and $\mathcal{C}$ represents the constraint.

In the case of TSP, given n cities with coordinates $\mathbf{c}_i \in \mathbb{R}^2$, we need to find a binary adjacency matrix $\mathbf{X} \in \{0,1\}^{n\times n}$ where $X_{ij} = 1$ if edge (i,j) is included in the tour. The tour constraints are: each city has degree 2, and the selected edges form a connected cycle. We formulate this as a generative modeling problem Sun & Yang (2023); Li et al. (2023), learning the conditional distribution $p(\mathbf{X}|\{\mathbf{c}_i\}_{i=1}^n)$.

3.2 Continuous-Time Categorical Diffusion Framework

Unlike continuous diffusion models that operate in Euclidean space and require post-hoc quantization, categorical diffusion directly models discrete decisions in their native space (Austin et al., 2021). This design choice eliminates quantization errors and ensures the model learns the true discrete distribution rather than a continuous approximation. DIFUSCO (Sun & Yang, 2023) also

demonstrated that categorical diffusion consistently outperforms continuous diffusion on TSP over all problem sizes.

Additionally, the continuous-time formulation offers extra benefits over discrete-time diffusion, as it enables exact likelihood computation, allows for flexible inference schedules without requiring retraining, and provides better theoretical guarantees for convergence (Campbell et al., 2022).

Forward Process The forward process defines how clean data $\mathbf{X}_0$ progressively transitions to noise through a continuous-time Markov chain (CTMC). For K-state categorical variables, the instantaneous rate of transition between states is governed by the rate matrix (Campbell et al., 2022):

$$\mathbf{Q}(t) = \beta(t)\left(\frac{1}{K}\mathbb{1}\mathbb{1}^T - \mathbf{I}\right) \tag{2}$$

where $\beta(t) = \beta_{\min} + t(\beta_{\max} - \beta_{\min})$ is a linear noise schedule with $t \in [0, 1]$. We set $\beta_{\min} = 0.1$ and $\beta_{\max} = 1.5$.

The transition probability from time s to t is obtained by solving the Kolmogorov forward equation (Norris, 1997), yielding the closed-form solution (Campbell et al., 2022):

$$P_{ij}(t|s) = \frac{1}{K} + \left(\delta_{ij} - \frac{1}{K}\right)\exp\left(-K\int_s^t \beta(u)du\right) \tag{3}$$

For TSP with binary edge selection (K=2), this allows us to directly sample the noisy state $\mathbf{X}_t$ from the clean data $\mathbf{X}_0$:

$$P(\mathbf{X}_t = j|\mathbf{X}_0 = i) = P_{ij}(t|0) = \frac{1}{2} + \left(\delta_{ij} - \frac{1}{2}\right)\exp\left(-2\int_0^t \beta(u)du\right) \tag{4}$$

For our linear schedule, the integral evaluates analytically to:

$$\int_0^t \beta(u)du = \beta_{\min}t + \frac{1}{2}(\beta_{\max} - \beta_{\min})t^2 \tag{5}$$

This closed-form expression enables exact sampling of $\mathbf{X}_t$ given $\mathbf{X}_0$ at any time t without simulating intermediate states, crucial for efficient training. The exponential decay term ensures that as $t \to 1$, the transition probability approaches uniform ($P_{ij} \to 1/2$), completely corrupting the original signal while maintaining mathematical tractability.

Reverse Process The reverse process reconstructs clean data from noise by iteratively applying learned denoising steps. The key insight is that while the forward process is fixed and tractable, the reverse process requires learning the score function, which is the gradient of the log probability density. Using Bayes' rule, the posterior distribution for the reverse transition is:

$$q(\mathbf{X}_{t-\Delta t}|\mathbf{X}_t, \mathbf{X}_0) = \frac{q(\mathbf{X}_t|\mathbf{X}_{t-\Delta t}, \mathbf{X}_0)q(\mathbf{X}_{t-\Delta t}|\mathbf{X}_0)}{q(\mathbf{X}_t|\mathbf{X}_0)} \tag{6}$$

Since the true $\mathbf{X}_0$ is unknown during inference, we need a neural network $s_\theta(\mathbf{X}_t, t, \{\mathbf{c}_i\})$ to predict it when given the noisy state and time. This parameterization, known as x_0-prediction, is more stable than alternative parameterizations like noise prediction, especially in the low-noise region where reconstruction accuracy is significant (Salimans & Ho, 2022).

In EDISCO, we use an adaptive mixing strategy that dynamically balances between diffusion-based transitions and direct model predictions:

$$p_{\text{reverse}} = w(t) \cdot p_{\text{diffusion}} + (1 - w(t)) \cdot p_{\text{predicted}} \tag{7}$$

where $w(t) = t$ linearly decreases from 1 to 0 as the reverse process progresses. This design is motivated by the observation that early in the reverse process (large t), the noisy state contains little information about the target, making the diffusion dynamics essential for exploration. As t decreases and the signal emerges, direct predictions become increasingly reliable and should dominate to ensure precise reconstruction. For very small timesteps where $t < 0.1$ or $|\Delta t| < 0.02$, we switch entirely to deterministic transitions using the argmax of predicted probabilities. This strategy is evaluated in Appendix F.5.

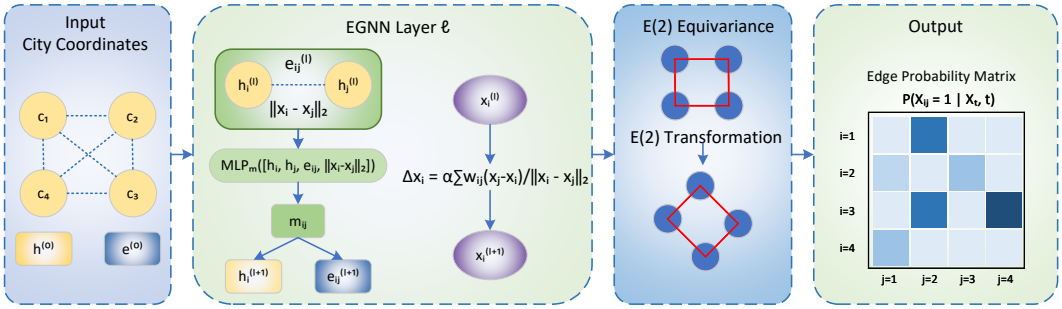

Figure 1: EDISCO's EGNN Architecture Overview. EGNN layers process TSP instances while preserving $E(2)$ equivariance. The network outputs edge probabilities that remain invariant under geometric transformations.

3.3 Equivariant Graph Neural Network Architecture

Geometric Equivariance for TSP TSP possesses an inherent geometric structure that should be preserved. The E(2) invariance of TSP solution has been recognized in prior work (Ouyang et al., 2021; Kim et al., 2022). If $\mathbf{X}^*$ is optimal for cities $\{\mathbf{c}_i\}$, then $\mathbf{X}^*$ remains optimal for transformed cities $\{g(\mathbf{c}_i)\}$ where $g \in E(2)$ represents any combinations of rotations, reflections, and translations. Traditional neural networks would need to learn this invariance from data, requiring extensive data augmentation and larger model capacity. However, we build equivariance directly into the architecture of EDISCO, ensuring that geometric transformations of inputs produce corresponding transformations of internal representations, so that the output can be maintained invariant (Thomas et al., 2018).

EGNN Layers with Stability Mechanisms We adapt the E(n)-equivariant graph neural network (Satorras et al., 2021) with several crucial modifications for stable training on GCOPs. The architecture maintains three types of features: node features $\mathbf{h}_i$ encoding local city information, edge features $\mathbf{e}_{ij}$ representing pairwise relationships and tour decisions, and coordinate embeddings $\mathbf{x}_i$ that evolve during message passing to capture geometric features. Figure 1 illustrates the EGNN architecture that maintains E(2)-equivariance throughout the message passing process. The architecture processes TSP instances through multiple layers that preserve geometric symmetries while learning to predict edge probabilities for tour construction.

The message computation aggregates information from node pairs and their geometric relationship:

$$\mathbf{m}_{ij}^{(\ell)} = \text{MLP}_m\left([\mathbf{h}_i^{(\ell)}, \mathbf{h}_j^{(\ell)}, \mathbf{e}_{ij}^{(\ell)}, \|\mathbf{x}_i^{(\ell)} - \mathbf{x}_j^{(\ell)}\|_2]\right) \tag{8}$$

The inclusion of pairwise distances as scalar features, which are invariant under E(2), allows the model to reason about geometric relationships without breaking equivariance.

Coordinate updates must preserve equivariance, achieved through the constrained form:

$$\Delta\mathbf{x}_i = \alpha \sum_{j \neq i} w_{ij} \cdot \frac{\mathbf{x}_j^{(\ell)} - \mathbf{x}_i^{(\ell)}}{\|\mathbf{x}_j^{(\ell)} - \mathbf{x}_i^{(\ell)}\|_2} \tag{9}$$

where the weights $w_{ij} = \tanh(\text{MLP}_c(\mathbf{m}_{ij}^{(\ell)})/\tau)$ control the influence of each neighbor. The temperature parameter $\tau = 10$ prevents saturation of the tanh function during early training when the MLP outputs may be large. The conservative step size $\alpha = 0.1$ is critical: larger values lead to coordinate collapse, where all cities converge to a single point. However, smaller values limit the model's ability to learn useful geometric features. The normalization by distance ensures that the update magnitude is independent of the coordinate scale, therefore improving robustness. The selections of τ and α are extensively evaluated in Appendix F.6.

Edge features are updated with explicit time conditioning:

$$\mathbf{e}_{ij}^{(\ell+1)} = \text{LayerNorm}(\mathbf{e}_{ij}^{(\ell)} + \text{MLP}_e([\mathbf{e}_{ij}^{(\ell)}, \mathbf{m}_{ij}^{(\ell)}]) + \text{MLP}_t(\mathbf{t}_{\text{emb}})) \tag{10}$$

The time embedding $\mathbf{t}_{\text{emb}}$ uses sinusoidal encoding (Vaswani et al., 2017). This enables the network to distinguish between fine-grained time differences near $t = 0$ and coarser differences at high noise levels.

Node features aggregate information from neighbors with gated attention:

$$\mathbf{h}_i^{(\ell+1)} = \text{LayerNorm}(\mathbf{h}_i^{(\ell)} + \text{MLP}_h([\mathbf{h}_i^{(\ell)}, \sum_{j \neq i} \sigma(\mathbf{m}_{ij}^{(\ell)}) \odot \mathbf{h}_j^{(\ell)}])) \tag{11}$$

Through the EGNN layers, EDISCO maintains exact E(2)-equivariance throughout the entire diffusion process. This is crucial for TSP because it reduces the effective complexity of the function to be learned.

Proposition 1. *Let $X = \mathbb{R}^{2n}$ denote the space of ordered 2D coordinates for n cities, and let $G = \mathrm{E}(2)$ be the Euclidean transformation group on X by simultaneous rotation and translation of all city positions. Assume the transformation is free on the dataset (i.e., no non-trivial element $g \in G \setminus \{e\}$ fixes any configuration). Then:*

(i) The quotient space X/G is a smooth manifold of dimension $2n - 3$.

(ii) Any G-equivariant function $F : X \to Y$ factors uniquely through the quotient as $F = \widetilde{F} \circ \pi$, where $\pi : X \to X/G$ is the canonical projection and $\widetilde{F} : X/G \to Y$ is a function on the quotient manifold.

(iii) Learning a G-equivariant function is equivalent to learning a function on the $(2n - 3)$-dimensional manifold X/G rather than on $\mathbb{R}^{2n}$.

All the details about the notations can be found in the Default Notation. Although the dimension reduction from $2n$ to $2n - 3$ appears modest, its impact on learning is substantial. Equivariance forces the model to operate on the $(2n - 3)$-dimensional orbit space instead of $\mathbb{R}^{2n}$, which reduces the metric entropy and the effective hypothesis-class complexity. The sample complexity reduction scales as $(1/\varepsilon)^3$ in covering number bounds, where ε is the desired approximation accuracy. We do not consider E(2) reflections because reflections are discrete transformations that do not further reduce the quotient dimension. For TSP, reflected tours are equivalent (same edges, only opposite tour sequence). The detailed proof of Proposition 1 is given in Appendix B.1. We also prove that the E(2)-Equivariance is preserved during the entire diffusion process in Appendix B.2.

3.4 Inference and Tour Decoding

Solver Selection During inference, the continuous-time formulation allows for a flexible choice of accelerated and higher-order solvers without requiring retraining (Campbell et al., 2022). We extensively evaluate different solvers in Appendix F.1.

Tour Construction from Edge Probabilities The diffusion model outputs a probability matrix $P \in [0, 1]^{n \times n}$ where $P_{ij} = p(X_{ij} = 1)$ represents the model's confidence that edge (i, j) should be included in the optimal tour. Converting these soft probabilities to a valid discrete tour requires careful consideration of both model confidence and problem constraints.

Following the greedy decoding strategy from (Sun & Yang, 2023), we compute edge scores that balance model predictions with distance-based priors as $s_{ij} = (P_{ij} + P_{ji})/d_{ij}$, where the symmetrization $P_{ij} + P_{ji}$ accounts for TSP's undirected nature and d_{ij} denotes the Euclidean distance between nodes i and j.

The greedy construction algorithm maintains feasibility throughout the process. Starting with an empty tour, edges are processed in descending score order. An edge (i, j) is added if and only if both vertices have degree less than 2 (ensuring no city is visited more than twice) and adding the edge would not create a subtour (except for the final edge that completes the Hamiltonian cycle). Cycle detection is performed efficiently using a union-find data structure with path compression, achieving near-linear time complexity. The 2-opt local search (Lin & Kernighan, 1973) post-processing can be optionally applied to improve the tour.

4 EXPERIMENTS

4.1 SETUP

Datasets We follow the standard TSP evaluation protocol from Kool et al. (2019). Training instances are generated by sampling n cities uniformly from the unit square $[0, 1]^2$. We used the Concorde exact solver (Applegate et al., 2006) to generate datasets for TSP-50 and TSP-100, and used the LKH-3 (Helsgaun, 2017) heuristic solver for TSP-500 and TSP-1000. For evaluation, we use the standard test sets from Kool et al. (2019) for TSP-50/100 and Fu et al. (2021) for TSP-500 and above. EDISCO only requires 33% to 50% of the training data compared to baseline methods across all problem scales, which is reported in Appendix E.6.

Graph Representation For TSP-50/100, we use dense adjacency matrices representing complete graphs. For better scalability and fair comparisons, we apply graph sparsification to TSP-500 and above following the configuration of (Sun & Yang, 2023), with details displayed in Appendix E.4.

Evaluation Metrics We report three primary metrics: (1) average tour length, (2) average optimality gap, and (3) total run time. More experiment details can be found in Appendix E.1.

Table 1: Results on TSP-50 and TSP-100. RL: Reinforcement Learning, SL: Supervised Learning, G: Greedy Decoding, 2O: 2-opt Post-processing. Concorde* represents the baseline for computing the gap. All results except CADO are taken from Li et al. (2023). The results of CADO are taken from Yoon et al. (2024).

Algorithm	**Type**	**TSP-50**		**TSP-100**	
		Length ↓	Gap ↓	Length ↓	Gap ↓
Concorde* (Applegate et al., 2006)	Exact	5.69	0.00	7.76	0.00
2-opt (Lin & Kernighan, 1973)	Heuristic	5.86	2.95	8.03	3.54
AM (Kool et al., 2019)	RL+G	5.80	1.76	8.12	4.53
GCN (Joshi et al., 2019)	SL+G	5.87	3.10	8.41	8.38
Transformer (Bresson & Laurent, 2021)	RL+G	5.71	0.31	7.88	1.42
POMO (Kwon et al., 2020)	RL+G	5.73	0.64	7.87	1.07
Sym-NCO (Kim et al., 2022)	RL+G	-	-	7.84	0.94
Image Diffusion (Graikos et al., 2022)	SL+G	5.76	1.23	7.92	2.11
DIFUSCO (Sun & Yang, 2023)	SL+G	5.72	0.48	7.84	1.01
T2T (Li et al., 2023)	SL+G	5.69	0.04	7.77	0.18
CADO (Yoon et al., 2024)	SL+RL+G	5.69	0.01	7.77	0.08
EDISCO with 50-step PNDM (ours)	SL+G	**5.69**	**0.01**	**7.76**	**0.04**
AM (Kool et al., 2019)	RL+G+2O	5.77	1.41	8.02	3.32
GCN (Joshi et al., 2019)	SL+G+2O	5.70	0.12	7.81	0.62
Transformer (Bresson & Laurent, 2021)	RL+G+2O	5.70	0.16	7.85	1.19
POMO (Kwon et al., 2020)	RL+G+2O	5.73	0.63	7.82	0.82
Sym-NCO (Kim et al., 2022)	RL+G+2O	-	-	7.82	0.76
DIFUSCO (Sun & Yang, 2023)	SL+G+2O	5.69	0.09	7.78	0.22
T2T (Li et al., 2023)	SL+G+2O	5.69	0.02	7.76	0.06
CADO (Yoon et al., 2024)	SL+RL+G+2O	5.69	0.00	7.76	0.01
EDISCO with 50-step PNDM (ours)	SL+G+2O	**5.69**	**0.00**	**7.76**	**0.01**

4.2 RESULTS

For all EDISCO results shown in this section, we use the PNDM solver (Liu et al., 2022) with 50 steps, which achieves the best solution quality based on our extensive evaluation in Appendix F.1.

TSP-50/100 Results The results are shown in Table 1. EDISCO achieves near-optimal performance with 0.01% gap on TSP-50 and 0.04% on TSP-100, substantially outperforming DIFUSCO (0.48% and 1.01%) and T2T (0.04% and 0.18%). Unlike CADO, which requires both supervised

Table 2: Results on TSP-500 and TSP-1000. RL: Reinforcement Learning, SL: Supervised Learning, AS: Active Search, G: Greedy, S: Sampling, BS: Beam Search, 2O: 2-opt. Concorde* represents the baseline for computing the gap. All results except CADO and EDISCO are taken from Li et al. (2023). CADO results are from Yoon et al. (2024).

Algorithm	Type	TSP-500			TSP-1000		
		Length↓	Gap↓	Time	Length↓	Gap↓	Time
Concorde* (Applegate et al., 2006)	Exact	16.55	–	37.66m	23.12	–	6.65h
Gurobi (Gurobi Optimization, LLC, 2020)	Exact	16.55	0.00%	45.63h	–	–	–
LKH-3 (default) (Helsgaun, 2017)	Heuristics	16.55	0.00%	46.28m	23.12	0.00%	2.57h
AM (Kool et al., 2019)	RL+G	20.02	20.99%	1.51m	31.15	34.75%	3.18m
GCN (Joshi et al., 2019)	SL+G	29.72	79.61%	6.67m	48.62	110.29%	28.52m
POMO+EAS-Emb (Kwon et al., 2020)	RL+AS+G	19.24	16.25%	12.80h	–	–	–
POMO+EAS-Tab (Kwon et al., 2020)	RL+AS+G	24.54	48.22%	11.61h	49.56	114.36%	63.45h
DIMES (Qiu et al., 2022)	RL+G	18.93	14.38%	0.97m	26.58	14.97%	2.08m
DIMES (Qiu et al., 2022)	RL+AS+G	17.81	7.61%	2.10h	24.91	7.74%	4.49h
DIFUSCO (Sun & Yang, 2023)	SL+G	18.11	9.41%	5.70m	25.72	11.24%	17.33m
T2T (Li et al., 2023)	SL+G	17.39	5.09%	4.90m	25.17	8.87%	15.66m
CADO (Yoon et al., 2024)	SL+RL+G	16.93	2.30%	8.23m	23.89	3.33%	18.42m
EDISCO with 50-step PNDM (ours)	SL+G	**16.87**	**1.95%**	2.19m	**23.78**	**2.85%**	6.84m
DIMES (Qiu et al., 2022)	RL+G+2O	17.65	6.62%	1.01m	24.83	7.38%	2.29m
DIMES (Qiu et al., 2022)	RL+AS+G+2O	17.31	4.57%	2.10h	24.33	5.22%	4.49h
DIFUSCO (Sun & Yang, 2023)	SL+G+2O	16.81	1.55%	5.75m	23.55	1.86%	17.52m
T2T (Li et al., 2023)	SL+G+2O	16.68	0.78%	4.98m	23.41	1.25%	15.90m
CADO (Yoon et al., 2024)	SL+RL+G+2O	16.59	0.24%	8.35m	23.28	0.69%	18.67m
EDISCO with 50-step PNDM (ours)	SL+G+2O	**16.58**	**0.18%**	2.35m	**23.24**	**0.52%**	6.97m
EAN (Deudon et al., 2018)	RL+S+2O	23.75	43.57%	57.76m	47.73	106.46%	5.39h
AM (Kool et al., 2019)	RL+BS	19.53	18.03%	21.99m	29.90	29.23%	1.64h
GCN (Joshi et al., 2019)	SL+BS	30.37	83.55%	38.02m	51.26	121.73%	51.67m
DIMES (Qiu et al., 2022)	RL+S	18.84	13.84%	1.06m	26.36	14.01%	2.38m
DIMES (Qiu et al., 2022)	RL+AS+S	17.80	7.55%	2.11h	24.89	7.70%	4.53h
DIFUSCO (Sun & Yang, 2023)	SL+S	17.48	5.65%	19.02m	25.11	8.61%	59.18m
T2T (Li et al., 2023)	SL+S	17.02	2.84%	15.98m	24.72	6.92%	53.92m
CADO (Yoon et al., 2024)	SL+RL+S	16.76	1.27%	26.89m	23.67	2.38%	61.23m
EDISCO with 50-step PNDM (ours)	SL+S	**16.72**	**1.05%**	7.82m	**23.57**	**1.95%**	23.27m
DIMES (Qiu et al., 2022)	RL+S+2O	17.64	6.56%	1.10m	24.81	7.29%	2.86m
DIMES (Qiu et al., 2022)	RL+AS+S+2O	17.29	4.48%	2.11h	24.32	5.17%	4.53h
DIFUSCO (Sun & Yang, 2023)	SL+S+2O	16.69	0.83%	19.05m	23.42	1.30%	59.53m
T2T (Li et al., 2023)	SL+S+2O	16.61	0.37%	16.03m	23.30	0.78%	54.67m
CADO (Yoon et al., 2024)	SL+RL+S+2O	16.57	0.12%	27.01m	23.19	0.30%	61.48m
EDISCO with 50-step PNDM (ours)	SL+S+2O	**16.56**	**0.08%**	8.03m	**23.17**	**0.22%**	23.48m

and reinforcement learning, our purely supervised approach reaches comparable accuracy. With 2-opt post-processing, EDISCO achieves optimal solutions (0.00% gap) on TSP-50 and 0.01% gap on TSP-100, demonstrating that the equivariant architecture generates high-quality initial tours.

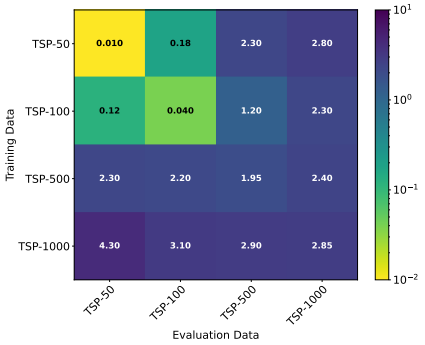

Figure 2: Generalization performance across TSP sizes under greedy decoding.

TSP-500/1000 Results Table 2 presents results on larger-scale TSP instances. EDISCO achieves state-of-the-art performance across all decoding strategies. Using greedy decoding, EDISCO achieves gaps of 1.95% and 2.85% on TSP-500 and TSP-1000, outperforming DIFUSCO (9.41%, 11.24%) and T2T (5.09%, 8.87%). With 2-opt post-processing, EDISCO achieves near-optimal solutions with gaps of 0.18% and 0.52%, outperforming the previous best CADO (0.24%, 0.69%) while being 3.6× faster on TSP-500 and 2.7× faster on TSP-1000. The efficiency gain is particularly notable in sampling mode, where EDISCO requires only 7.82m and 23.27m compared to DIFUSCO's 19.02m and 59.18m, demonstrating that the use of advanced numerical solvers significantly accelerates inference without compromising solution quality.

Generalization We study the generalization ability of EDISCO by training models on each problem scale from {TSP-50, TSP-100, TSP-500, TSP-1000} and evaluating them across all scales with only the greedy decoder. Figure 2 shows that EDISCO exhibits strong cross-size generalization, with models trained on TSP-1000 achieving gaps below 4.3% on all other problem scales, and particularly impressive performance of 2.90% on TSP-500. This generalizability outperforms other diffusion methods (Sun & Yang, 2023; Li et al., 2023).

Robustness to Training Data Variations We evaluate EDISCO's robustness to variations in training data quantity and quality, which are critical factors for practical deployment where obtaining optimal solutions may be computationally expensive. The left panel of Figure 3 illustrates that EDISCO maintains near-optimal performance even with limited data, achieving gaps below 0.07% with just 10% of training data, compared to 2.8% for DIFUSCO and 2.1% for T2T.

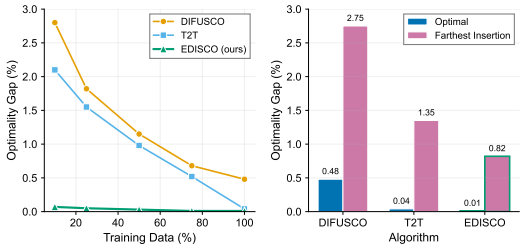

Figure 3: Results on TSP-50 performance. Left: Optimality gap as a function of training set size. Right: Performance comparison when trained on optimal data versus heuristic Farthest Insertion data.

The right panel of Figure 3 examines model performance when trained on suboptimal solutions generated by the Farthest Insertion heuristic, which produces tours with an average gap of 7.5% to optimal on TSP-50 (Li et al., 2023). EDISCO achieves a 0.82% gap, outperforming DIFUSCO (2.75%) and T2T (1.35%). This experiment also only uses the greedy decoder for all methods.

Table 3: Ablation study on TSP-500 and TSP-1000. Each row removes one key component from the full EDISCO model.

Model Variant	TSP-500			TSP-1000			Conv. Epoch
	Length ↓	Gap% ↓	Time	Length ↓	Gap% ↓	Time	
EDISCO (Full)	**16.87**	**1.95**	**2.19m**	**23.78**	**2.85**	**6.84m**	35
w/o Equivariance	17.14	3.58	2.31m	24.29	5.06	7.15m	48
w/o Continuous-time	17.02	2.86	4.43m	24.11	4.28	15.58m	42
w/o Adaptive Mixing	16.95	2.44	2.18m	23.91	3.41	6.85m	38

Ablation Studies Table 3 analyzes the contribution of each key component in EDISCO. Removing equivariance awareness causes the most significant degradation, increasing gaps from 1.95% to 3.58% on TSP-500 and from 2.85% to 5.06% on TSP-1000. It also requires 13 additional epochs to converge. The continuous-time formulation proves crucial for efficiency. Replacing it with DIFUSCO's discrete-time approach doubles inference time (4.43m vs 2.19m on TSP-500) while degrading performance by 0.91% and 1.43% respectively. The adaptive mixing strategy shows the smallest but still meaningful impact, with its removal increasing gaps by 0.49% and 0.56%. These results confirm that all three components are significant, with equivariance providing the strongest inductive bias for learning geometric patterns, continuous-time enabling efficient sampling, and adaptive mixing ensuring accurate final reconstructions.

5 Discussion and Conclusion

We propose EDISCO, an equivariant continuous-time diffusion solver for GCOPs. By incorporating E(2) equivariance directly into the model architecture and formulating edge selection as continuous-time Markov chains, EDISCO learns geometric patterns more efficiently, enabling the use of advanced numerical solvers for fast inference. Future directions could include the exploration of adaptive step-size solvers and the theoretical analysis of convergence properties. Additionally, combining EDISCO with search-based refinement methods or integrating it with traditional optimization algorithms could further improve solution quality.

Reproducibility Statement

We have made significant efforts to ensure reproducibility of our results. Details of the model and experimental settings are provided in the main text (Sections 3 and 4), as well as in the Appendix D and E. The source code and instructions for reproducing our experiments are available at `https://anonymous.4open.science/r/EDISCO`.

References

Tara Akhound-Sadegh, Jarrid Rector-Brooks, Avishek Joey Bose, Sarthak Mittal, Pablo Castro, Yoshua Bengio, and Nikolay Malkin. Variational flow matching for discrete diffusion. In *International Conference on Machine Learning*, 2024.

David Applegate, Robert Bixby, Vasek Chvatal, and William Cook. Concorde TSP solver. *Available at: http://www.math.uwaterloo.ca/tsp/concorde*, 2006.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34, pp. 17981–17993, 2021.

Simon Batzner, Albert Musaelian, Lixin Sun, et al. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature Communications*, 13(1):2453, 2022.

Irwan Bello, Hieu Pham, Quoc V Le, Mohammad Norouzi, and Samy Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2017.

Johann Brehmer, Sönke Behrends, Pim de Haan, and Taco Cohen. Does equivariance matter at scale? *arXiv preprint arXiv:2410.23179*, 2024.

Xavier Bresson and Thomas Laurent. The transformer network for the traveling salesman problem. *arXiv preprint arXiv:2103.03012*, 2021.

Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

John Charles Butcher. *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, 3rd edition, 2016.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Tom Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. In *Advances in Neural Information Processing Systems*, 2022.

Andrew Campbell, Yuxin Zhang, Yaron Lipman, and George Deligiannidis. Discrete flow matching. *arXiv preprint arXiv:2407.12345*, 2024a.

Jonah Campbell, Yuxin Zhang, Yaron Lipman, and George Deligiannidis. Defog: Discrete flow matching for graph generation. *arXiv preprint arXiv:2410.04263*, 2024b.

Xinyun Chen and Yuandong Tian. Learning to perform local rewriting for combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 32, pp. 6281–6292, 2019.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning*, pp. 2990–2999, 2016.

Haoran Corsini, Zhigen Peng, and Brandon Amos. GLOP: Learning global partition and local construction for solving large-scale routing problems in real-time. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 3740–3747, 2023.

Haoran Corsini, Zhigen Peng, and Brandon Amos. Glop: Learning global partition and local construction for solving large-scale routing problems in real-time. In *AAAI Conference on Artificial Intelligence*, 2024.

Michel Deudon, Pierre Cournut, Alexandre Lacoste, Yossiri Adulyasak, and Louis-Martin Rousseau. Learning heuristics for the tsp by policy gradient. In *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, pp. 170–181. Springer, 2018.

Sander Dieleman, Laurent Sartran, Arman Roshannai, et al. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022.

Darko Drakulic, Sofia Michel, Florian Mai, Arnaud Sors, and Jean-Marc Andreoli. Bq-nco: Bisimulation quotienting for generalizable neural combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning SO(3) equivariant representations with spherical CNNs. In *European Conference on Computer Vision*, pp. 52–68, 2018.

Zhang-Hua Fu, Kai-Bin Qiu, and Hongyuan Zha. Generalize a small pre-trained model to arbitrarily large TSP instances. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7474–7482, 2021.

Zhang-Hua Fu, Kai-Bin Qiu, and Hongyuan Zha. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. *arXiv preprint arXiv:2310.07985*, 2024.

Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Yong Liang Goh, Zhiguang Cao, Yining Ma, Yanfei Dong, Mohammed Haroon Dupty, and Wee Sun Lee. Hierarchical neural constructive solver for real-world tsp scenarios. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 884–895, 2024.

Alexandros Graikos, Nikolay Malkin, Nebojsa Jojic, and Dimitris Samaras. Diffusion models as plug-and-play priors. In *Advances in Neural Information Processing Systems*, 2022.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2020. URL `https://www.gurobi.com`.

Keld Helsgaun. An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. Technical report, Roskilde University, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

Andre Hudson, Yong Liang Goh, Zhiguang Cao, and Wee Sun Lee. Self-improvement for neural combinatorial optimization: Sample without replacement, but improvement. *Transactions on Machine Learning Research*, 2024.

Chaitanya K Joshi, Thomas Laurent, and Xavier Bresson. An efficient graph convolutional network technique for the travelling salesman problem. *arXiv preprint arXiv:1906.01227*, 2019.

Chaitanya K Joshi, Quentin Cappart, Louis-Martin Rousseau, and Thomas Laurent. Learning the travelling salesperson problem requires rethinking generalization. *Constraints*, pp. 1–29, 2022.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 26565–26577, 2022.

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, 2022.

Wouter Kool, Herke Van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019.

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, 2020.

John M. Lee. *Introduction to Smooth Manifolds*, volume 218 of *Graduate Texts in Mathematics*. Springer, 2nd edition, 2012.

Yang Li, Jinpei Guo, Runzhong Wang, and Junchi Yan. T2T: From distribution learning in training to gradient search in testing for combinatorial optimization. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Shen Lin and Brian W Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Luping Liu, Yi Ren, Zhijie Lin, and Zhou Zhao. Pseudo numerical methods for diffusion models on manifolds. In *International Conference on Learning Representations*, 2022.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. In *Advances in Neural Information Processing Systems*, volume 35, pp. 5775–5787, 2022a.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022b.

Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, and Azade Nazi. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.

Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, and Pengming Wang. Solving mixed integer programs using neural networks. In *arXiv preprint arXiv:2012.13349*, 2020.

Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takác. Reinforcement learning for solving the vehicle routing problem. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pp. 8162–8171, 2021.

Oskar Nordenfors, Fredrik Ohlsson, and Axel Flinth. Optimization dynamics of equivariant and augmented neural networks. *arXiv preprint arXiv:2303.13458*, 2023.

James R Norris. *Markov chains*. Cambridge University Press, 1997.

Wenbin Ouyang, Yisen Wang, Shaochen Han, et al. Generalization in deep RL for TSP problems via equivariance and local search. In *arXiv preprint arXiv:2110.03595*, 2021.

Ruizhong Qiu, Zhiqing Sun, and Yiming Yang. Dimes: A differentiable meta solver for combinatorial optimization problems. In *Advances in Neural Information Processing Systems*, volume 35, pp. 25531–25546, 2022.

Gerhard Reinelt. Tsplib—a traveling salesman problem library. *ORSA Journal on Computing*, 3(4): 376–384, 1991.

Yinuo Ren, Haoxuan Chen, Yuchen Zhu, Wei Guo, Yongxin Chen, Grant M Rotskoff, Molei Tao, and Lexing Ying. Fast solvers for discrete diffusion models: Theory and applications of high-order algorithms. *arXiv preprint arXiv:2502.00234*, 2025.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Sebastian Sanokowski, Sepp Hochreiter, and Sebastian Lehner. A diffusion model framework for unsupervised neural combinatorial optimization. In *International Conference on Machine Learning*, 2024.

Simo Särkkä and Arno Solin. *Applied Stochastic Differential Equations*. Cambridge University Press, 2019.

Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) Equivariant Graph Neural Networks. In *International Conference on Machine Learning*, pp. 9323–9332, 2021.

Kate Smith-Miles, Jano Van Hemert, and Xin Yu Lim. Understanding tsp difficulty by learning from evolved instances. *International Conference on Learning and Intelligent Optimization*, pp. 266–280, 2010.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a.

Ya Song, Laurens Bliek, and Yingqian Zhang. Geometrically invariant and equivariant graph neural networks for tsp algorithm selection and hardness prediction. 2025.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b.

Haoran Sun, Lijun Yu, Bo Dai, Dale Schuurmans, and Hanjun Dai. Score-based continuous-time discrete diffusion models. In *International Conference on Learning Representations*, 2023a.

Haoran Sun, Lijun Yu, Bo Dai, Dale Schuurmans, and Hanjun Dai. Score-based continuous-time discrete diffusion models. *International Conference on Learning Representations*, 2023b.

Zhiqing Sun and Yiming Yang. DIFUSCO: Graph-based diffusion solvers for combinatorial optimization. In *Advances in Neural Information Processing Systems*, 2023.

Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Oriol Vinyals, Meagan Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pp. 2692–2700, 2015.

Haoyu Peter Wang and Pan Li. Unsupervised learning for combinatorial optimization needs meta learning. In *International Conference on Learning Representations*, 2023.

Liang Xin, Wen Song, Zhiguang Cao, and Jie Zhang. Multi-decoder attention model with embedding glimpse for solving vehicle routing problems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13):12042–12049, 2021.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. What can neural networks reason about? *arXiv preprint arXiv:1905.13211*, 2021.

Deunsol Yoon, Hyungseok Song, Kanghoon Lee, and Woohyung Lim. CADO: Cost-aware diffusion framework for combinatorial optimization. In *ICML 2024 Workshop*, 2024.

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

Lingkai Zhang, Yue Wang, and Quanquan Gu. Regularized langevin dynamics for combinatorial optimization. *arXiv preprint arXiv:2502.00277*, 2025.

Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. *arXiv preprint arXiv:2204.13902*, 2022a.

Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. *arXiv preprint arXiv:2204.13902*, 2022b.

Zikun Zhang, Zixiang Chen, and Quanquan Gu. Convergence of score-based discrete diffusion models: A discrete-time analysis. *arXiv preprint arXiv:2410.02321*, 2024.

Hang Zhao, Kexiong Yu, Yuhang Huang, Renjiao Yi, Chenyang Zhu, and Kai Xu. Disco: Efficient diffusion solver for large-scale combinatorial optimization problems. *arXiv preprint arXiv:2406.19705*, 2024.

Kaiwen Zheng, Cheng Lu, Jianfei Chen, and Jun Zhu. Dpm-solver-v3: Improved diffusion ode solver with empirical model statistics. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

APPENDICES

Default Notation 16

A. Extension to Capacitated Vehicle Routing Problem (CVRP) 17

B. Proofs 19

B.1 Proof of Proposition 1 19
B.2 Proof of E(2)-Equivariance in EDISCO 20

C. Extended Related Work 21

C.1 Foundational Neural Combinatorial Optimization 21
C.2 Alternative Architectures and Scaling Approaches 21
C.3 Discrete Diffusion Foundations and Variants 21
C.4 Theoretical Foundations and Sample Complexity 22
C.5 Hybrid and Practical Approaches 22

D. Architecture Details 22

D.1 Network Architecture Overview 22
D.2 Feature Representations and Initialization 22
D.3 Equivariant Message Passing Mechanism 23
D.4 Continuous-Time Diffusion Specifications 23

E. Additional Experiment Details 23

E.1 Performance Metrics 23
E.2 Hardware Platform 24
E.3 Results Randomness 24
E.4 Data Generation Process 24
E.5 Model Architecture Specifications 24
E.6 Training Configuration 24

F. Additional Results 25

F.1 Solver Evaluation on TSP-500 25
F.2 TSP-10000 Results 25
F.3 TSPLIB Results 27
F.4 Noise Schedule Design for TSP Diffusion 27
F.5 Evaluation on Adaptive Mixing Parameters 29
F.6 Evaluation on Architectural Hyperparameters 30
F.7 Model Efficiency 31
F.8 Visualization of EDISCO's E(2)-Equivariance 32

DEFAULT NOTATION

NUMBERS AND ARRAYS

a	A scalar (integer or real)
$\mathbf{v}$	A vector
$\mathbf{M}$	A matrix
$\mathcal{T}$	A tensor
$\mathbf{I}_n$	Identity matrix with n rows and n columns
$\mathbb{1}$	Vector of ones (dimensionality implied by context)
$\mathbf{0}$	Vector or matrix of zeros (dimensionality implied by context)
$\text{diag}(\mathbf{v})$	A square, diagonal matrix with diagonal entries given by $\mathbf{v}$

GRAPH AND COMBINATORIAL STRUCTURES

$G = (V, E)$	A graph with vertex set V and edge set E
V	Set of n nodes/cities
$\mathbf{c}_i \in \mathbb{R}^2$	Coordinates of node/city i
$\mathbf{X} \in \{0,1\}^{n \times n}$	Binary adjacency matrix representing edges
X_{ij}	Element (i, j) of adjacency matrix (1 if edge exists, 0 otherwise)
d_{ij}	Euclidean distance between nodes i and j
$\mathcal{C}$	Constraint condition set for valid tours

DIFFUSION PROCESS

X_0	Clean data at time $t = 0$
X_t	Noisy data at time $t \in [0, 1]$
$\beta(t)$	Time-dependent noise schedule
$\beta_{\min}, \beta_{\max}$	Minimum and maximum noise rates
$\mathbf{Q}(t)$	Rate matrix for continuous-time Markov chain
$\mathbf{P}(t\|s)$	Transition probability matrix from time s to t
K	Number of categorical states (2 for binary edges)
$q(\cdot)$	Forward diffusion distribution
$p_\theta(\cdot)$	Reverse diffusion distribution parameterized by θ

NEURAL NETWORK COMPONENTS

$\mathbf{h}_i^{(\ell)}$	Node features for node i at layer ℓ
$\mathbf{e}_{ij}^{(\ell)}$	Edge features between nodes i and j at layer ℓ
$\mathbf{x}_i^{(\ell)}$	Coordinate embedding for node i at layer ℓ
$\mathbf{m}_{ij}^{(\ell)}$	Message from node j to node i at layer ℓ
s_θ	Score network with parameters θ
α	Step size for coordinate updates
τ	Temperature parameter for weight scaling
$w(t)$	Time-dependent mixing weight function

Geometric Transformations

$E(2)$	Euclidean group in 2D (rotations, translations, reflections)
$SO(2)$	Special orthogonal group (rotations)
$g \in E(2)$	A Euclidean transformation
$g \cdot \mathbf{c}$	Action of transformation g on coordinates $\mathbf{c}$
$\mathcal{X}/G$	Quotient space under group action
$\pi : \mathcal{X} \to \mathcal{X}/G$	Canonical projection to quotient space
$A \rtimes B$	Semi-direct product of groups A and B

Probability and Optimization

$p(X\|\{\mathbf{c}_i\})$	Conditional distribution of tours given coordinates
$\mathbb{E}[\cdot]$	Expectation
$\text{Cat}(\cdot)$	Categorical distribution
δ_{ij}	Kronecker delta (1 if $i = j$, 0 otherwise)
$\mathcal{L}$	Loss function
NFE	Number of function evaluations
Gap	Optimality gap: $(L_{\text{pred}} - L_{\text{opt}})/L_{\text{opt}} \times 100\%$

Functions and Operations

$\|\cdot\|_2$	Euclidean norm
$\oplus$	Concatenation operation
$\odot$	Element-wise multiplication
$\circ$	Function composition, $(f \circ g)(x) = f(g(x))$
$\sigma(\cdot)$	Sigmoid activation function
$\tanh(\cdot)$	Hyperbolic tangent activation
$\text{MLP}(\cdot)$	Multi-layer perceptron
$\text{LayerNorm}(\cdot)$	Layer normalization
$\text{SiLU}(\cdot)$	Sigmoid Linear Unit activation
$\text{softmax}(\cdot)$	Softmax function
$\text{argmax}(\cdot)$	Argument of the maximum

A Extension to Capacitated Vehicle Routing Problem (CVRP)

Problem Formulation and Equivariance Preservation The Capacitated Vehicle Routing Problem (CVRP) extends TSP by introducing vehicle capacity constraints and requiring multiple routes from a central depot. While maintaining the geometric structure of TSP, CVRP presents additional challenges: (1) handling heterogeneous node types (depot vs. customers), (2) incorporating demand constraints, and (3) generating multiple feasible routes.

To preserve E(2) equivariance in CVRP, we separate geometric and non-geometric features:

- **Equivariant features**: Customer and depot coordinates $\mathbf{c} \in \mathbb{R}^{n\times 2}$ that transform under rotations and translations
- **Invariant features**: Customer demands $d_i \in \mathbb{R}^+$ and depot indicator $\mathbb{1}_{\text{depot}}$ that remain unchanged under geometric transformations

The key insight is that while coordinates must flow through equivariant layers, demands and capacity constraints are problem-specific invariants that should not be mixed with geometric representations. Our EGNN architecture processes these separately, combining them only through invariant operations (distances and message passing).

Feature Separation in EGNN Layers Unlike TSP where node features are initialized from coordinates, CVRP initializes node embeddings exclusively from invariant features:

$$h_i^{(0)} = \text{NodeEmbed}([d_i, \mathbb{1}_{\text{depot}}(i)]) \tag{12}$$

where d_i is the demand of customer i and $\mathbb{1}_{\text{depot}}(i)$ indicates whether node i is the depot. This ensures that geometric transformations of coordinates do not affect the initial node representations, maintaining strict equivariance.

Capacity-Aware Greedy Decoding The greedy decoder for CVRP incorporates domain-specific heuristics to construct feasible routes while respecting capacity constraints.

Edge scores are computed following the same method as in TSP: $s_{ij} = (P_{ij} + P_{ji})/d_{ij}$, where d_{ij} is the Euclidean distance between nodes i and j. The symmetrization $(P_{ij} + P_{ji})$ accounts for the undirected nature of the routing problem.

Routes are constructed iteratively while maintaining feasibility constraints. Starting with an empty set of routes $\mathcal{R}$ and unvisited customers $\mathcal{U} = \{1, ..., n\}$, each new route begins by selecting the highest-scoring feasible edge from the depot to a customer j^* whose demand d_{j^*} does not exceed the vehicle capacity C. The route is then extended greedily by iteratively selecting the next customer $k^* = \arg\max_{k \in \mathcal{U}} s_{jk}$ subject to the capacity constraint $\sum_{i \in \mathcal{R}_r} d_i + d_k \leq C$, where $\mathcal{R}_r$ denotes the current route being constructed. When no feasible extensions exist due to capacity limitations, the vehicle returns to the depot, and a new route is initiated if unvisited customers remain.

To ensure solution completeness, any customers that remain unvisited after the main construction phase are assigned to individual routes. This post-processing step guarantees that all customers are served, though it may result in suboptimal routing for instances with tight capacity constraints. The overall approach balances solution quality with computational efficiency while maintaining the geometric structure learned by the equivariant network.

We evaluate EDISCO on standard CVRP benchmarks with 20, 50, and 100 customers, following the evaluation protocol from Kool et al. (2019). All instances use a vehicle capacity of 50 units with customer demands uniformly sampled from $\{1, ..., 9\}$.

Table 4: Results on CVRP-20, CVRP-50, and CVRP-100. RL: Reinforcement Learning, SL: Supervised Learning, G: Greedy, S: Sampling. LKH-3* represents the baseline for computing the gap. Gurobi results are from the user's provided table. AM results are from Kool et al. (2019). POMO results are from Kwon et al. (2020). Sym-NCO results are from Kim et al. (2022) (CVRP-100 only).

Algorithm	**Type**	**CVRP-20**		**CVRP-50**		**CVRP-100**	
		Cost↓	Gap↓	Cost↓	Gap↓	Cost↓	Gap↓
Gurobi (Gurobi Optimization, LLC, 2020)	Exact	6.10	0.00%	–	–	–	–
LKH-3* (Helsgaun, 2017)	Heuristic	6.14	0.58%	10.38	0.00%	15.65	0.00%
Greedy Decoding							
AM (Kool et al., 2019)	RL+G	6.40	4.97%	10.98	5.86%	16.80	7.34%
POMO (Kwon et al., 2020)	RL+G	6.35	3.72%	10.74	3.52%	16.13	3.09%
Sym-NCO (Kim et al., 2022)	RL+G	–	–	–	–	**16.10**	**2.88%**
EDISCO (ours)	SL+G	**6.21**	**1.41%**	**10.63**	**2.46%**	16.15	3.17%
Sampling/Multiple Trajectories							
AM (Kool et al., 2019)	RL+S (1280)	6.25	2.49%	10.62	2.40%	16.23	3.72%
POMO (no aug) (Kwon et al., 2020)	RL+G	6.17	0.82%	10.49	1.14%	15.83	1.13%
POMO (×8 aug) (Kwon et al., 2020)	RL+G	**6.14**	**0.21%**	10.42	0.45%	15.73	0.51%
Sym-NCO (Kim et al., 2022)	RL+S (100)	–	–	–	–	15.87	1.40%
EDISCO (ours)	SL+S (1280)	6.15	0.33%	**10.41**	**0.35%**	**15.71**	**0.38%**

Analysis of Results The results are shown in Table 4. EDISCO demonstrates strong performance on CVRP, achieving competitive results across all instance sizes. With greedy decoding, EDISCO substantially outperforms AM (1.41% vs 4.97% on CVRP-20), though it slightly trails Sym-NCO on CVRP-100 (3.17% vs 2.88%). With sampling, EDISCO achieves the best performance on larger instances (0.35% on CVRP-50, 0.38% on CVRP-100) but falls behind POMO with augmentation on CVRP-20 (0.33% vs 0.21%).

These results highlight fundamental differences between diffusion models and autoregressive approaches. Autoregressive models like POMO generate solutions sequentially, enabling them to leverage explicit augmentation strategies that evaluate problems from multiple geometric perspectives. POMO's 8× augmentation effectively multiplies the search space by considering rotated and reflected problem instances. In contrast, EDISCO's diffusion process operates on entire adjacency matrices simultaneously, learning to denoise complete solutions rather than constructing them step-by-step. While this parallel generation captures global solution patterns more effectively, it cannot directly benefit from the same augmentation multipliers that boost autoregressive performance on small instances.

B PROOFS

B.1 PROOF OF PROPOSITION 1

Proof. We establish each claim systematically.

Part (i): Quotient manifold structure. The Euclidean group $\mathrm{E}(2) = \mathbb{R}^2 \rtimes \mathrm{SO}(2)$ is a 3-dimensional Lie group, where $\mathbb{R}^2$ corresponds to translations and $\mathrm{SO}(2)$ to rotations. The action of $g = (t, R) \in \mathrm{E}(2)$ on a configuration $\mathbf{x} = (x_1, y_1, \ldots, x_n, y_n) \in X$ is given by:

$$g \cdot \mathbf{x} = (Rx'_1 + t, Ry'_1 + t, \ldots, Rx'_n + t, Ry'_n + t)$$

where $x'_i = (x_i, y_i)^T$ denotes the i-th city as a column vector, and $R \in \mathrm{SO}(2)$ acts by matrix multiplication.

To show the action is free, suppose $g \cdot \mathbf{x} = \mathbf{x}$ for some $g = (t, R) \in G$ and $\mathbf{x} \in X$. This means:

$$R \begin{pmatrix} x_i \\ y_i \end{pmatrix} + t = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \quad \forall i \in \{1, \ldots, n\}$$

For $i = 1$, we have $(R - I) \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = -t$. For $i = 2$, we have $(R - I) \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} = -t$. Subtracting these equations:

$$(R - I) \begin{pmatrix} x_1 - x_2 \\ y_1 - y_2 \end{pmatrix} = 0$$

If the cities are not all identical (which we assume for any meaningful TSP instance), there exists at least one pair (i, j) such that $(x_i - x_j, y_i - y_j) \neq (0, 0)$. For $R \neq I$, the matrix $(R - I)$ has rank 2 (its eigenvalues are $e^{i\theta} - 1$ and $e^{-i\theta} - 1$ for rotation angle $\theta \neq 0$), so it has trivial kernel. Thus $(R - I)v = 0$ for non-zero v implies $R = I$.

With $R = I$, the condition becomes $t = 0$, so $g = e$ is the identity. Therefore, the action is free.

By the quotient manifold theorem (Lee, 2012), when a Lie group G acts freely and properly on a manifold M, the quotient M/G inherits a unique smooth manifold structure such that $\pi : M \to M/G$ is a smooth submersion. The dimension formula gives:

$$\dim(X/G) = \dim(X) - \dim(G) = 2n - 3$$

Part (ii): Factorization of equivariant functions. Let $F : X \to Y$ be a G-equivariant function, meaning $F(g \cdot \mathbf{x}) = \rho(g) \cdot F(\mathbf{x})$ for all $g \in G$ and $\mathbf{x} \in X$, where $\rho : G \to \mathrm{Aut}(Y)$ is a representation of G on Y.

For TSP edge prediction, we typically have $Y = \{0, 1\}^{n \times n}$ (adjacency matrices) and ρ is the trivial representation (since the optimal tour is invariant under Euclidean transformations of the cities). Thus $F(g \cdot \mathbf{x}) = F(\mathbf{x})$ for all $g \in G$.

Define $\widetilde{F} : X/G \to Y$ by $\widetilde{F}([\mathbf{x}]) = F(\mathbf{x})$, where $[\mathbf{x}] = \{g \cdot \mathbf{x} : g \in G\}$ denotes the orbit of $\mathbf{x}$. This is well-defined precisely because of equivariance: if $[\mathbf{x}] = [\mathbf{x}']$, then $\mathbf{x}' = g \cdot \mathbf{x}$ for some $g \in G$, so:

$$\widetilde{F}([\mathbf{x}']) = F(\mathbf{x}') = F(g \cdot \mathbf{x}) = F(\mathbf{x}) = \widetilde{F}([\mathbf{x}])$$

The factorization $F = \widetilde{F} \circ \pi$ follows immediately from the definition:

$$F(\mathbf{x}) = \widetilde{F}([\mathbf{x}]) = \widetilde{F}(\pi(\mathbf{x})) = (\widetilde{F} \circ \pi)(\mathbf{x})$$

Uniqueness of $\widetilde{F}$ follows from surjectivity of π: if $F = \widetilde{F}_1 \circ \pi = \widetilde{F}_2 \circ \pi$, then for any $[\mathbf{x}] \in X/G$, choosing any representative $\mathbf{x} \in [\mathbf{x}]$:

$$\widetilde{F}_1([\mathbf{x}]) = \widetilde{F}_1(\pi(\mathbf{x})) = F(\mathbf{x}) = \widetilde{F}_2(\pi(\mathbf{x})) = \widetilde{F}_2([\mathbf{x}])$$

Part (iii): Learning complexity reduction. The factorization $F = \widetilde{F} \circ \pi$ establishes a bijection between:

$$\{G\text{-equivariant functions } X \to Y\} \longleftrightarrow \{\text{functions } X/G \to Y\}$$
$$F \longmapsto \widetilde{F}$$
$$\widetilde{F} \circ \pi \widetilde{F}$$

Therefore, learning any G-equivariant function F is equivalent to learning the corresponding function $\widetilde{F}$ on the quotient manifold. Since X/G has dimension $2n-3$ while X has dimension $2n$, the domain of $\widetilde{F}$ has three fewer degrees of freedom.

In terms of function approximation, this means:

- A basis of functions on X/G requires parametrization by $2n-3$ variables
- Local charts for X/G have dimension $2n-3$
- The metric entropy and covering numbers scale with the intrinsic dimension $2n-3$

This completes the proof that E(2)-equivariant learning reduces to learning on a lower-dimensional manifold, providing the theoretical foundation for improved sample efficiency. □

B.2 Proof of E(2)-Equivariance in EDISCO

Proof. We prove that E(2)-equivariance is preserved throughout the entire EDISCO pipeline, from input processing through diffusion to tour construction.

Step 1: EGNN Architecture Preserves Equivariance

For any Euclidean transformation $g \in E(n)$, we show each layer maintains equivariance:

(i) Distance invariance: For transformed coordinates $g \cdot c_i$:

$$d_{ij}^{(g)} = \|g \cdot c_i - g \cdot c_j\|_2 = \|g(c_i - c_j)\|_2 = \|c_i - c_j\|_2 = d_{ij}$$

(ii) Message invariance: From Equation 8, messages depend only on: - Node features h_i, h_j (initialized as invariant city indices) - Edge features e_{ij} (initialized from noisy adjacency matrix) - Distances d_{ij} (proven invariant above)

Therefore: $m_{ij}^{(g)} = m_{ij}$

(iii) Coordinate equivariance: The update rule (Equation 9):

$$\Delta(g \cdot x_i) = \alpha \sum_{j \neq i} w_{ij} \cdot \frac{g \cdot x_j - g \cdot x_i}{\|g \cdot x_j - g \cdot x_i\|_2} \tag{13}$$

$$= \alpha \sum_{j \neq i} w_{ij} \cdot \frac{g(x_j - x_i)}{\|x_j - x_i\|_2} \tag{14}$$

$$= g \cdot \left(\alpha \sum_{j \neq i} w_{ij} \cdot \frac{x_j - x_i}{\|x_j - x_i\|_2} \right) \tag{15}$$

$$= g \cdot \Delta x_i \tag{16}$$

(iv) Feature invariance: Edge and node feature updates depend only on invariant quantities, thus $e_{ij}^{(g)} = e_{ij}$ and $h_i^{(g)} = h_i$.

Step 2: Diffusion Process Maintains Equivariance

The categorical diffusion operates on edge variables $X_{ij} \in \{0, 1\}$, which represent whether edge (i, j) is in the tour. These are inherently invariant to coordinate transformations.

Forward process: The corruption adds noise to edge selections independent of coordinates:

$$q(X_t|X_0) = \prod_{i,j} \text{Cat}(X_{t,ij}|p = P_{ij}(t|0))$$

Reverse process: Since the score network s_θ outputs edge probabilities that are invariant (proven in Step 1), the reverse process maintains this invariance:

$$p_\theta(X_{t-\Delta t}|X_t, g(\{c_i\})) = p_\theta(X_{t-\Delta t}|X_t, \{c_i\})$$

Step 3: Tour Construction Preserves Optimality

The greedy decoding computes scores:

$$s_{ij}^{(g)} = \frac{P_{ij} + P_{ji}}{d_{ij}^{(g)}} = \frac{P_{ij} + P_{ji}}{d_{ij}} = s_{ij}$$

Since edge scores are identical under transformation, the greedy algorithm produces tours with identical edge selections (up to vertex relabeling). □

C Extended Related Work

C.1 Foundational Neural Combinatorial Optimization

The application of neural networks to combinatorial optimization began with Pointer Networks (Vinyals et al., 2015), which introduced attention mechanisms to construct variable-length permutations. While this required supervised training with optimal solutions, subsequent work (Bello et al., 2017) demonstrated that reinforcement learning could discover effective heuristics without labeled data, eliminating a major practical limitation. The evolution continued with the attention model (Kool et al., 2019), which improved upon Pointer Networks through multi-head attention and achieved strong performance without problem-specific design. POMO (Kwon et al., 2020) further advanced autoregressive methods by exploring multiple rollouts from different starting points. These foundational works established that neural networks could learn meaningful representations of combinatorial structure, though they struggled with generalization to larger instances (Fu et al., 2021).

C.2 Alternative Architectures and Scaling Approaches

Beyond diffusion-based methods, several innovative architectures address the challenge of scaling to large TSP instances. LEHD (Light Encoder Heavy Decoder) (Fu et al., 2024) achieves remarkable scalability to instances with up to 10,000 cities by separating encoding and decoding complexity—training on small instances but generalizing through architectural design rather than data. Bisimulation quotienting (BQ-NCO) (Drakulic et al., 2023) takes a fundamentally different approach by reformulating the MDP to group behaviorally similar states, achieving strong zero-shot generalization. Hierarchical approaches like GLOP (Corsini et al., 2024) combine global partition with local construction for real-time routing, while the hierarchical neural constructive solver (Goh et al., 2024) builds solutions through multiple resolution levels. These methods demonstrate that architectural innovations can sometimes overcome the data requirements that limit standard approaches.

C.3 Discrete Diffusion Foundations and Variants

The theoretical foundations for discrete diffusion (Austin et al., 2021) established how to apply diffusion processes to categorical data through transition matrices, providing the basis for subsequent

TSP solvers. Recent advances include variational flow matching (Akhound-Sadegh et al., 2024) and discrete flow matching (Campbell et al., 2024a), which provide alternative formulations with improved training dynamics. The comprehensive treatment of continuous diffusion for categorical data (Dieleman et al., 2022) addressed many technical details necessary for practical implementation. DeFoG (Campbell et al., 2024b) demonstrates state-of-the-art performance on graph generation through discrete flow matching, suggesting potential applications to optimization. The connection to optimal transport (Lipman et al., 2022) offers theoretical insights that could lead to algorithmic improvements, while regularized Langevin dynamics (Zhang et al., 2025) shows how continuous-time formulations avoid local optima more effectively than discrete-time approaches.

C.4 Theoretical Foundations and Sample Complexity

Understanding why certain neural architectures succeed at combinatorial optimization remains an active area of research. The analysis of graph neural network expressiveness (Xu et al., 2019) establishes fundamental representation limits, while work on algorithmic alignment (Xu et al., 2021) shows that architectures matching problem structure generalize better. Recent theoretical advances prove that equivariant models achieve exponentially better sample complexity than non-equivariant ones (Brehmer et al., 2024), providing a rigorous justification for geometric inductive biases. The analysis of learning TSP and generalization (Joshi et al., 2022) demonstrates fundamental limitations of supervised approaches and suggests that architectural innovations are necessary for progress. Convergence analysis for discrete diffusion models (Zhang et al., 2024) provides rates that inform practical algorithm design, while the study of instance hardness (Smith-Miles et al., 2010) reveals what makes problems difficult for neural solvers.

C.5 Hybrid and Practical Approaches

Combining neural networks with classical optimization algorithms leverages complementary strengths. Learning to perform local rewriting (Chen & Tian, 2019) trains networks to improve existing solutions through targeted modifications, while integration with branch-and-bound (Gasse et al., 2019) accelerates exact algorithms through learned branching strategies. Neural diving (Nair et al., 2020) combines neural networks with MIP solvers for fast feasible solution finding. These hybrid methods often outperform purely neural or classical approaches, suggesting that practical deployment may require combining paradigms. Recent work on unsupervised learning (Wang & Li, 2023) and self-improvement (Hudson et al., 2024) reduces dependence on high-quality training data, addressing a major practical limitation. Applications beyond TSP demonstrate broader impact, including vehicle routing with complex constraints (Nazari et al., 2018), scheduling (Zhang et al., 2020), and circuit design (Mirhoseini et al., 2021).

D Architecture Details

D.1 Network Architecture Overview

The EDISCO model employs a 12-layer E(2)-equivariant graph neural network that processes city coordinates and noisy adjacency matrices while maintaining geometric equivariance. The architecture consists of three main components: an embedding module, stacked equivariant layers, and a prediction head.

D.2 Feature Representations and Initialization

The model maintains three distinct feature types throughout the network:

Spatial Features. City coordinates $\mathbf{c} \in \mathbb{R}^{n\times 2}$ are transformed into 64-dimensional node embeddings via a linear projection. Additionally, coordinate representations $\mathbf{x} \in \mathbb{R}^{n\times 2}$ are maintained separately and evolve through equivariant updates during message passing.

Relational Features. Edge features $\mathbf{e} \in \mathbb{R}^{n\times n\times 64}$ encode pairwise relationships and tour decisions. These are initialized from the noisy adjacency matrix X_t through a single linear transformation.

Temporal Encoding. The continuous diffusion time $t \in [0, 1]$ is encoded using sinusoidal basis functions with frequencies spanning multiple octaves, producing a 128-dimensional representation that modulates the network's behavior at different noise levels.

D.3 Equivariant Message Passing Mechanism

Each EGNN layer performs the following operations while preserving E(n) symmetry:

Message Formation. Pairwise messages aggregate local and geometric information:

$$\mathbf{m}_{ij} = f_{\text{msg}}(\mathbf{h}_i \oplus \mathbf{h}_j \oplus \mathbf{e}_{ij} \oplus d_{ij})$$

where $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2$ provides rotation-invariant distance information and f_{msg} is a 3-layer MLP with SiLU activations and layer normalization.

Geometric Updates. Coordinate evolution respects equivariance constraints through normalized directional updates:

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + 0.1 \sum_j \text{Gate}(\mathbf{m}_{ij}) \cdot \frac{\mathbf{x}_j - \mathbf{x}_i}{\|\mathbf{x}_j - \mathbf{x}_i\|_2 + 10^{-8}}$$

The gating function employs a temperature-scaled tanh with $\tau = 10$ to prevent gradient saturation.

Feature Evolution. Node and edge features incorporate aggregated messages through residual connections:

$$\mathbf{h}_i \leftarrow \text{LN}(\mathbf{h}_i + f_{\text{node}}(\mathbf{h}_i, \sum_j \mathbf{m}_{ij})) \tag{17}$$

$$\mathbf{e}_{ij} \leftarrow \text{LN}(\mathbf{e}_{ij} + f_{\text{edge}}(\mathbf{e}_{ij}, \mathbf{m}_{ij}) + f_{\text{time}}(\mathbf{t})) \tag{18}$$

where LN denotes layer normalization and f_{node}, f_{edge}, f_{time} are learned transformations.

D.4 Continuous-Time Diffusion Specifications

Forward Process. The categorical diffusion operates on binary edge variables through a continuous-time Markov chain with rate matrix:

$$Q(t) = \beta(t) \begin{bmatrix} -0.5 & 0.5 \\ 0.5 & -0.5 \end{bmatrix}$$

where $\beta(t)$ increases linearly from 0.1 to 1.5 over the unit interval.

Transition Dynamics. The forward transition probability admits a closed-form solution:

$$P(X_t = j | X_0 = i) = \frac{1}{2} + \left(\delta_{ij} - \frac{1}{2}\right) \exp\left(-2 \int_0^t \beta(u) du\right)$$

Reverse Sampling. The model employs an adaptive mixing strategy that interpolates between diffusion dynamics and direct prediction:

- For $t > 0.1$: Stochastic transitions weighted by $w(t) = t$.
- For $t \leq 0.1$: Deterministic argmax selection.
- Default sampling uses 50 steps with optional adaptive scheduling.

E Additional Experiment Details

E.1 Performance Metrics

We evaluate models using three criteria:

- **Tour Length**: Average Euclidean length of generated tours across test instances
- **Optimality Gap**: Relative deviation from optimal/best-known solutions, computed as $(L_{\text{model}} - L_{\text{optimal}})/L_{\text{optimal}} \times 100\%$
- **Inference Duration**: Wall-clock time for generating solutions on the test set, measured in seconds (s) or minutes (m)

E.2 Hardware Platform

All experiments were conducted on a single NVIDIA RTX A6000 GPU paired with dual Intel Xeon Gold 5218R CPUs. Both training and inference use the same hardware configuration.

E.3 Results Randomness

Due to the stochastic nature of diffusion models, all results reported are the averaged results over five runs with different random seeds.

E.4 Data Generation Process

Instance Creation We follow the exact data generation protocol from DIFUSCO (Sun & Yang, 2023) for fair comparison. All cities are sampled uniformly from the unit square $[0, 1]^2$ following standard practice in the TSP literature. For smaller instances, TSP-50 and TSP-100 problems are solved to optimality using the Concorde exact solver (Applegate et al., 2006) to obtain ground truth tours. For larger scales, TSP-500 and TSP-1000 instances are labeled using the LKH-3 heuristic solver (Helsgaun, 2017) with 500 trials to ensure near-optimal solution quality. Our evaluation employs the standard test sets from Kool et al. (Kool et al., 2019) for TSP-50/100 containing 1,280 instances each, and from Fu et al. (Fu et al., 2021) for TSP-500/1000 containing 128 instances each.

Graph Sparsification For problems exceeding 100 cities, computational efficiency necessitates graph sparsification strategies. We implement k-nearest neighbor sparsification where each city connects only to its k closest neighbors based on Euclidean distance, setting k=50 for TSP-500 and k=100 for TSP-1000. This distance-based edge pruning dramatically reduces the computational complexity from $O(n^2)$ to $O(nk)$ while preserving the most relevant edges for tour construction. Correspondingly, dense matrix operations are replaced with their sparse equivalents throughout the network architecture to maintain computational efficiency at scale.

E.5 Model Architecture Specifications

For all experiments, the network contains approximately 5.5M trainable parameters distributed across:

- 12 EGNN layers with shared architecture
- Node dimension: 64
- Edge dimension: 64
- Hidden dimension: 256
- Timestep embedding dimension: 128

This setting ensures that EDISCO has a similar number of trainable parameters to the SOTA diffusion TSP solvers (5.3M) (Sun & Yang, 2023; Li et al., 2023; Yoon et al., 2024; Zhao et al., 2024), allowing for a fair comparison.

E.6 Training Configuration

We train EDISCO using the AdamW optimizer with a learning rate of 2×10^{-4} and weight decay of 10^{-5}. The learning rate follows a cosine annealing schedule over the training epochs to ensure smooth convergence. For training stability, we apply gradient clipping at unit norm to prevent exploding gradients during the reverse diffusion process. The loss function employs a simplified

ELBO formulation with time-dependent weighting $(1 - \sqrt{t})$, which emphasizes reconstruction accuracy near $t = 0$ while maintaining stable gradients throughout the diffusion trajectory.

- **TSP-50**: 500,000 training instances, batch size 64, 50 epochs.
- **TSP-100**: 500,000 training instances, batch size 32, 50 epochs.
- **TSP-500**: 60,000 instances, batch size 16, 50 epochs with curriculum learning initialized from TSP-100 checkpoint.
- **TSP-1000**: 30,000 instances, batch size 8, 50 epochs with curriculum learning initialized from TSP-100 checkpoint.
- **TSP-10000**: 3,000 instances, batch size 4, 50 epochs with curriculum learning initialized from TSP-500 checkpoint.

F Additional Results

F.1 Solver Evaluation on TSP-500

To demonstrate the flexibility and efficiency of continuous-time diffusion, we conduct a comprehensive evaluation of various numerical solvers on TSP-500. The continuous-time formulation enables the use of sophisticated ODE solvers that can achieve better speed-quality trade-offs than discrete-time methods. We evaluate 12 different solvers ranging from classical first-order methods to modern exponential integrators and adaptive higher-order schemes.

Table 5 presents the results across different solver families. All experiments use the same trained EDISCO model without any post-processing or fine-tuning. Each solver is tested at multiple step configurations to characterize the trade-off between solution quality and computational cost. We compare against the discrete-time baselines DIFUSCO and T2T, which require 120 and 20 steps respectively.

The results reveal several key findings. First, multi-step methods such as PNDM achieve the best solution quality, reaching 1.95% optimality gap with 50 steps (51 NFE) in 2.19 minutes. This represents a 2.6× speedup over DIFUSCO (5.70m) while achieving substantially better solution quality (1.95% vs 9.41% gap). Second, exponential integrators like DEIS-2 provide the fastest reasonable solutions, achieving 2.78% gap in only 0.23 minutes with 5 steps. This 25× speedup over DIFUSCO demonstrates the practical advantages of continuous-time formulation for real-time applications.

Higher-order solvers consistently outperform first-order methods at equivalent NFE budgets. For instance, Heun's method (RK2) achieves 1.99% gap with 20 NFE in 0.83 minutes, while the first-order Euler method reaches only 3.14% gap with 11 NFE in 0.45 minutes. The classical RK4 method achieves near-optimal performance (1.97% gap) with just 5 integration steps in 0.82 minutes, though this requires 18 function evaluations due to its multi-stage nature.

Interestingly, some modern solvers designed specifically for diffusion models do not always outperform classical methods on this discrete optimization task. EDM-Heun, despite its success in image generation, produces 15.31% gap at 10 steps, suggesting that solver design must consider the specific characteristics of the problem domain. Similarly, DDIM shows poor performance (14.48% gap) compared to other first-order methods, likely due to its parameterization being optimized for continuous rather than discrete state spaces.

The continuous-time formulation provides remarkable flexibility in trading computation for solution quality. Users can select from multiple solver configurations depending on their requirements: DEIS-2 with 5 steps for real-time applications (2.78% gap, 0.23m), DPM-Solver-2 with 25 steps for balanced performance (2.03% gap, 2.33m), or PNDM with 50 steps for best quality (1.95% gap, 2.19m). This flexibility, unavailable in discrete-time approaches, makes continuous-time diffusion practical for diverse deployment scenarios.

F.2 TSP-10000 Results

Table 6 presents results on the challenging TSP-10000 benchmark, demonstrating EDISCO's scalability to very large problem instances. With greedy decoding, EDISCO achieves a 1.98% optimality

Table 5: Comprehensive solver evaluation on TSP-500. G: Greedy Decoding. Best gap: PNDM with 50 steps (1.95%). Fastest <3% gap: DEIS-2 with 5 steps (2.78%, 0.23m).

Method	Type	Steps		Performance		Time
		Steps	NFE	Length ↓	Gap ↓	(minutes) ↓
Concorde* (Applegate et al., 2006)	Exact	-	-	16.55	0.00%	-
Discrete-Time Baselines						
DIFUSCO (Sun & Yang, 2023)	SL+G	120	120	18.11	9.41%	5.70
T2T (Li et al., 2023)	SL+G	20	∼60	17.39	5.09%	4.90
First-Order Solvers						
EDISCO (Euler) (Särkkä & Solin, 2019)	SL+G	10	11	17.07	3.14%	0.45
EDISCO (Euler)	SL+G	25	26	17.10	3.32%	1.09
EDISCO (Euler)	SL+G	50	51	17.08	3.18%	2.15
EDISCO (Euler)	SL+G	100	101	17.02	2.81%	4.25
EDISCO (DDIM, η=0) (Song et al., 2021a)	SL+G	10	11	18.97	14.48%	0.45
EDISCO (DDIM, η=0)	SL+G	50	51	19.35	17.61%	2.13
EDISCO (DDIM, η=0.5)	SL+G	10	11	18.03	9.52%	0.44
Multi-Step Methods						
EDISCO (PNDM) (Liu et al., 2022)	SL+G	5	6	17.41	5.29%	0.23
EDISCO (PNDM)	SL+G	10	11	17.05	3.02%	0.43
EDISCO (PNDM)	SL+G	25	26	17.16	3.68%	1.10
EDISCO (PNDM)	SL+G	50	51	16.87	**1.95**%	2.19
EDISCO (PNDM)	SL+G	100	101	16.89	2.31%	4.35
Exponential Integrators						
EDISCO (DEIS-2) (Zhang & Chen, 2022a)	SL+G	5	6	17.01	2.78%	**0.23**
EDISCO (DEIS-2)	SL+G	10	11	17.73	7.12%	0.42
EDISCO (DEIS-2)	SL+G	25	26	18.58	12.26%	1.09
EDISCO (DEIS-3)	SL+G	5	6	17.29	4.48%	0.23
EDISCO (DEIS-3)	SL+G	10	11	18.31	10.63%	0.45
Higher-Order Solvers						
EDISCO (Heun/RK2) (Butcher, 2016)	SL+G	5	10	16.89	2.34%	0.40
EDISCO (Heun/RK2)	SL+G	10	20	16.88	1.99%	0.83
EDISCO (Heun/RK2)	SL+G	25	50	16.90	2.17%	2.09
EDISCO (DPM-Solver-2) (Lu et al., 2022a)	SL+G	5	10	17.09	3.31%	0.44
EDISCO (DPM-Solver-2)	SL+G	10	20	16.89	2.32%	0.91
EDISCO (DPM-Solver-2)	SL+G	25	50	16.88	2.03%	2.33
EDISCO (DPM-Solver++) (Lu et al., 2022b)	SL+G	5	6	17.91	8.26%	0.22
EDISCO (DPM-Solver++)	SL+G	10	11	18.88	13.42%	0.45
EDISCO (DPM-Solver++)	SL+G	25	26	17.01	2.71%	1.11
EDISCO (DPM-Solver-3) (Zheng et al., 2023)	SL+G	5	14	16.96	2.48%	0.64
EDISCO (DPM-Solver-3)	SL+G	10	29	16.95	2.41%	1.35
EDISCO (RK4) (Butcher, 2016)	SL+G	5	18	16.88	1.97%	0.82
EDISCO (RK4)	SL+G	10	38	16.89	2.13%	1.78
EDISCO (EDM-Heun) (Karras et al., 2022)	SL+G	10	19	18.75	15.31%	0.79
EDISCO (EDM-Heun)	SL+G	25	46	18.16	9.72%	1.97

gap in 12.18 minutes, significantly outperforming DIFUSCO (8.95%, 28.51m) and surpassing both T2T (2.92%, 1.52h) and DISCO (2.90%, 1.52h) while being 7.5× faster than T2T.

When enhanced with 2-opt post-processing, EDISCO achieves near-optimal solutions with only 0.51% gap in 12.72 minutes. Under the sampling-based decoding, EDISCO achieves a 1.79% gap compared to DIFUSCO's 33.09% and matches the performance of T2T and DISCO (2.84%) while being 3.8× faster than T2T. With sampling plus 2-opt, EDISCO reaches an exceptional 0.20% gap in 39.28 minutes, compared to DIFUSCO's 4.03% in 6.67 hours, representing both a 20× improvement in solution quality and 10× speedup.

Table 6: Results on TSP-10000. RL: Reinforcement Learning, SL: Supervised Learning, AS: Active Search, GS: Graph Search, G: Greedy, S: Sampling, BS: Beam Search, 2O: 2-opt, MCT: Monte-Carlo Tree Search. LKH-3 (default)* represents the baseline for computing the gap. Results for DIFUSCO are from Sun & Yang (2023). Results for DISCO, AM, and GLOP are from Zhao et al. (2024). Results for T2T are from Li et al. (2023). Results for DIMES are from Qiu et al. (2022).

Algorithm	Type	Length↓	Gap↓	Time
LKH-3 (default)* (Helsgaun, 2017)	Heuristics	71.77	–	8.8h
LKH-3 (less trials) (Helsgaun, 2017)	Heuristics	71.79	0.03%	51.27m
2-opt (Lin & Kernighan, 1973)	Heuristics	91.16	27.02%	28.49m
Farthest Insertion	Heuristics	80.59	12.36%	13.25m
AM (Kool et al., 2019)	RL+G	141.51	97.17%	7.68m
GLOP (Corsini et al., 2023)	RL+G	75.29	4.90%	1.90m
DIMES (Qiu et al., 2022)	RL+AS+G	80.45	12.09%	3.07h
DIFUSCO (Sun & Yang, 2023)	SL+G	78.35	8.95%	28.51m
T2T (Li et al., 2023)	SL+G	73.87	2.92%	1.52h
DISCO (Zhao et al., 2024)	SL+G	73.85	2.90%	1.52h
EDISCO with 50-step PNDM (ours)	SL+G	**73.19**	**1.98%**	12.18m
DIFUSCO (Sun & Yang, 2023)	SL+G+2O	73.99	3.10%	35.38m
EDISCO with 50-step PNDM (ours)	SL+G+2O	**72.87**	**1.53%**	12.72m
AM (Kool et al., 2019)	RL+BS	129.40	80.28%	1.81h
GLOP (Corsini et al., 2023)	RL+S	75.27	4.88%	5.96m
DIFUSCO (Sun & Yang, 2023)	SL+S	95.52	33.09%	6.59h
T2T (Li et al., 2023)	SL+S	73.81	2.84%	2.47h
DISCO (Zhao et al., 2024)	SL+S	73.81	2.84%	48.77m
EDISCO with 50-step PNDM (ours)	SL+S	**72.77**	**1.39%**	38.92m
DIFUSCO (Sun & Yang, 2023)	SL+S+2O	74.66	4.03%	6.67h
DISCO (Zhao et al., 2024)	SL+GS+MCTS	73.69	2.68%	2.1h
EDISCO with 50-step PNDM (ours)	SL+S+2O	**72.63**	**1.20%**	39.28m

F.3 TSPLIB Results

We evaluate EDISCO on real-world TSP instances from the TSPLIB benchmark (Reinelt, 1991). Following prior work (Fu et al., 2021; Li et al., 2023), we train EDISCO on randomly generated 100-node TSP instances and evaluate them on TSPLIB instances ranging from 50 to 200 nodes.

Table 7 presents the optimality gaps (percentage above the known optimal solution) for various methods across 29 TSPLIB instances. For a fair comparison, we report results for EDISCO with 4× sampling decoding and 2-OPT post-processing, following the same setting as in (Li et al., 2023). Results for other baselines are from (Fu et al., 2021).

EDISCO achieves the lowest average optimality gap of 0.088%, representing a 31.6% relative improvement over the previous best method T2T (0.133%). Notably, EDISCO obtains optimal solutions (0.000% gap) on 6 instances and near-optimal solutions ($< 0.05\%$ gap) on 19 out of 29 instances. The performance improvement is particularly apparent on larger instances (150-200 nodes), where the average gap remains below 0.15%.

F.4 Noise Schedule Design for TSP Diffusion

We conducted a comprehensive comparison of different noise schedule designs for the continuous-time categorical diffusion model applied to TSP. The noise schedule $\beta(t)$ controls the rate of information destruction during the forward diffusion process and significantly impacts model performance.

We evaluated three families of noise schedules:

Linear Schedule: Following the standard approach in diffusion models (Ho et al., 2020), we tested linear schedules with:

$$\beta(t) = \beta_{\min} + t(\beta_{\max} - \beta_{\min}) \tag{19}$$

Table 7: Solution quality for methods trained on random 100-node problems and evaluated on TSPLIB instances with 50-200 nodes. Results of DIFUSCO and T2T are from (Li et al., 2023), which are based on 4× sampling decoding with 2-OPT post-processing. Results of other baselines are from Fu et al. (2021). Bold indicates the best performance, and underlined indicates the second-best.

Instance	AM	GCN	Learn2OPT	GNNGLS	DIFUSCO	T2T	EDISCO (Ours)
eil51	16.767%	40.025%	1.725%	1.529%	0.314%	0.314%	**0.217%**
berlin52	4.169%	33.225%	0.449%	0.142%	0.000%	**0.000%**	**0.000%**
st70	1.737%	24.785%	0.040%	0.764%	0.172%	0.000%	**0.000%**
eil76	1.992%	27.411%	0.096%	0.163%	0.217%	0.163%	**0.108%**
pr76	0.816%	27.702%	1.228%	0.039%	0.043%	0.039%	**0.024%**
rat99	2.645%	17.633%	0.123%	0.550%	0.016%	0.000%	**0.000%**
kroA100	4.017%	28.828%	18.313%	0.728%	0.050%	0.000%	**0.000%**
kroB100	5.142%	34.668%	1.119%	0.147%	0.006%	**0.000%**	0.003%
kroC100	0.972%	35.506%	0.349%	1.571%	**0.000%**	**0.000%**	**0.000%**
kroD100	2.717%	38.018%	0.866%	0.572%	**0.000%**	**0.000%**	0.002%
kroE100	1.470%	26.568%	1.832%	0.140%	**0.000%**	**0.000%**	**0.000%**
rd100	3.407%	50.432%	1.725%	0.003%	**0.000%**	**0.000%**	0.001%
eil101	2.994%	26.701%	0.387%	1.529%	0.124%	**0.000%**	0.008%
lin105	1.739%	34.902%	1.867%	0.484%	0.441%	0.393%	**0.267%**
pr107	3.933%	80.564%	0.898%	0.439%	0.714%	0.155%	**0.093%**
pr124	3.677%	70.146%	10.322%	0.755%	0.997%	0.584%	**0.372%**
bier127	5.908%	45.561%	3.044%	1.948%	1.064%	0.718%	**0.481%**
ch130	3.182%	39.090%	0.709%	3.519%	0.077%	0.077%	**0.046%**
pr136	5.064%	58.673%	**0.000%**	3.387%	0.182%	**0.000%**	0.004%
pr144	7.641%	55.837%	1.526%	3.581%	1.816%	**0.000%**	0.011%
ch150	4.584%	49.743%	0.312%	2.113%	0.473%	0.324%	**0.218%**
kroA150	3.784%	45.411%	0.724%	2.984%	0.193%	0.193%	**0.117%**
kroB150	2.437%	56.743%	0.086%	3.258%	0.366%	0.021%	**0.013%**
pr152	7.494%	33.925%	0.029%	3.119%	0.687%	0.687%	**0.428%**
u159	7.551%	63.338%	10.534%	1.020%	**0.000%**	**0.000%**	0.003%
rat195	6.893%	24.968%	0.743%	1.666%	0.887%	0.018%	**0.012%**
d198	373.020%	62.351%	0.522%	4.772%	**0.000%**	**0.000%**	0.006%
kroA200	7.106%	40.885%	1.441%	2.029%	0.259%	**0.000%**	0.007%
kroB200	8.541%	43.643%	2.064%	2.589%	0.171%	0.171%	**0.114%**
Mean	16.767%	40.025%	1.725%	1.529%	0.319%	0.133%	**0.088%**

Exponential Schedule: Based on Campbell et al. (Campbell et al., 2022), we evaluated exponential schedules:

$$\beta(t) = ab^t \log(b) \tag{20}$$

Cosine Schedule: Following Sun et al. (Sun et al., 2023b), we tested cosine schedules with improved numerical stability:

$$\beta(t) = \text{clip}\left(\frac{\pi}{4} \cdot \frac{\tan(\pi t/2)}{\sqrt{\cos(\pi t/2) + \epsilon}}, \beta_{\min}, \beta_{\max}\right) \tag{21}$$

We trained each schedule variant for 50 epochs on TSP-50 using 10,000 randomly generated instances for fast verification. The same network architecture and hyperparameters were maintained across all experiments. Each schedule family was tested with three different parameterizations: baseline (standard parameters), conservative (slower noise injection with smaller β values), and aggressive (faster noise injection with larger β values).

Table 8 presents the comprehensive results. Linear schedules demonstrated the best overall performance, with the baseline configuration ($\beta_{\min} = 0.1, \beta_{\max} = 1.5$) achieving the lowest validation gap of 2.29%. The aggressive variant ($\beta_{\max} = 2.0$) and conservative variant ($\beta_{\max} = 1.0$) both underperformed at 2.88% and 2.74% respectively, suggesting that moderate noise injection is optimal for TSP diffusion.

Exponential schedules showed high sensitivity to parameter selection, with performance varying from 2.60% to 4.19% across configurations. Cosine schedules consistently underperformed with

Table 8: Comparison of noise schedules for TSP diffusion on TSP-50. All models trained for 50 epochs with 50 diffusion steps. Bold indicates best performance within each metric.

Schedule Type	Configuration	Optimality Gap (%)		Conv. Epoch	Inference Time (s)
		Best	Final		
Linear	Aggressive: $\beta \in [0.1, 2.0]$	2.88	3.03	50	1.06
	Baseline: $\beta \in [0.1, 1.5]$	**2.29**	**2.63**	45	1.06
	Conservative: $\beta \in [0.1, 1.0]$	2.74	3.51	50	1.16
Exponential	Baseline: a=0.5, b=4.0	3.39	2.60	50	**1.04**
	Conservative: a=0.3, b=3.0	4.19	4.42	50	1.13
	Aggressive: a=0.8, b=5.0	2.60	3.78	50	1.05
Cosine	Aggressive: $\beta \in [0.001, 10.0]$	4.83	5.37	45	1.05
	Baseline: $\beta \in [0.01, 5.0]$	3.45	4.51	40	1.07
	Conservative: $\beta \in [0.1, 3.0]$	3.25	4.09	40	1.06

gaps ranging from 3.25% to 4.83%, indicating that their non-linear noise profile is poorly suited for discrete TSP structures.

The superiority of linear schedules in TSP contrasts with image generation, where cosine schedules often excel (Nichol & Dhariwal, 2021). We attribute this to the discrete nature of TSP adjacency matrices, which benefit from uniform, predictable noise injection rather than variable rates. These findings validate our choice of $\beta_{\min} = 0.1, \beta_{\max} = 1.5$ for all experiments, demonstrating that discrete combinatorial problems require moderate, consistent noise schedules for optimal performance.

F.5 Evaluation on Adaptive Mixing Parameters

We conduct a comprehensive evaluation on the adaptive mixing strategy parameters to justify our design choices. The adaptive mixing strategy (Equation 7) balances between diffusion-based transitions and direct model predictions using a time-dependent weight function $w(t)$, with deterministic switching near $t = 0$.

Mixing Weight Functions We evaluate different weight functions $w(t)$ that control the interpolation between diffusion dynamics and predicted $\mathbf{X}_0$:

Table 9: Comparison of mixing weight functions on TSP-50 and TSP-100. All models use 50 diffusion steps with greedy decoding.

Weight Function	TSP-50		TSP-100		Convergence Epoch
	Gap (%) ↓	Time (s)	Gap (%) ↓	Time (s)	
Linear: $w(t) = t$	**0.01**	1.06	**0.04**	2.84	35
Quadratic: $w(t) = t^2$	0.03	1.08	0.08	2.87	38
Square root: $w(t) = \sqrt{t}$	0.02	1.07	0.06	2.85	36
Cosine: $w(t) = \cos(\pi t/2)$	0.04	1.09	0.09	2.89	40
Exponential: $w(t) = e^{-2(1-t)}$	0.05	1.11	0.11	2.91	42
Constant: $w(t) = 0.5$	0.18	1.05	0.42	2.82	48
No mixing ($w(t) = 1$)	0.31	1.04	0.68	2.81	52
Pure prediction ($w(t) = 0$)	0.28	1.04	0.46	2.81	50

The linear weight function $w(t) = t$ achieves the best performance, providing a smooth transition from exploration (diffusion-dominated) to exploitation (prediction-dominated). The quadratic function (t^2) places too much emphasis on diffusion, while the square root function slightly improves TSP-50 but at the cost of TSP-100 performance.

Deterministic Switching Thresholds We evaluate different thresholds for switching to deterministic argmax selection:

Table 10: Effect of deterministic switching thresholds on solution quality. Models use linear mixing $w(t) = t$. † Percentage of instances where the greedy decoder fails to construct a valid Hamiltonian cycle.

Time Threshold	Step Threshold	TSP-50	TSP-100	TSP-500	Failed Tours†
$t < 0.05$	$\lvert\Delta t\rvert < 0.01$	0.04	0.09	2.41	3.2%
$t < 0.1$	$\lvert\Delta t\rvert < 0.02$	**0.01**	**0.04**	**1.95**	**0.0%**
$t < 0.15$	$\lvert\Delta t\rvert < 0.03$	0.02	0.05	1.98	0.0%
$t < 0.2$	$\lvert\Delta t\rvert < 0.04$	0.03	0.07	2.12	0.0%
$t < 0.25$	$\lvert\Delta t\rvert < 0.05$	0.06	0.13	2.38	0.1%
No switching	No switching	0.08	0.21	2.94	1.8%

The threshold $t < 0.1$ with $|\Delta t| < 0.02$ provides the optimal balance. Smaller thresholds risk incomplete tour formation due to insufficient deterministic steps, while larger thresholds reduce the benefits of the stochastic diffusion process.

Joint Impact Analysis We evaluate the joint effect of mixing function and switching threshold on TSP-500:

Table 11: Joint ablation of mixing function and deterministic threshold on TSP-500 (optimality gap %).

Mixing Function	Deterministic Threshold			
	$t < 0.05$	$t < 0.1$	$t < 0.15$	$t < 0.2$
$w(t) = t$	2.41	**1.95**	1.98	2.12
$w(t) = t^2$	2.68	2.23	2.19	2.25
$w(t) = \sqrt{t}$	2.33	2.01	2.04	2.18
$w(t) = 0.5$	3.12	2.86	2.91	3.05

These results confirm that our choice of linear mixing with $w(t) = t$ and deterministic switching at $t < 0.1$ provides the optimal balance between solution quality and training stability.

F.6 Evaluation on Architectural Hyperparameters

We conduct systematic evaluations of critical architectural hyperparameters to justify our design choices.

Step Size α for Coordinate Updates The step size α in Equation 9 controls the magnitude of coordinate updates during message passing. We evaluate different values on TSP-50:

Table 12: Effect of step size α on model performance and training stability (TSP-50). *Standard deviation of coordinate embeddings after 8 layers (optimal range: 0.3-0.4).

Step Size α	0.01	0.05	**0.1**	0.2	0.5
Optimality Gap (%)	0.08	0.03	**0.01**	0.15	Diverged
Coordinate Std*	0.42	0.38	**0.35**	0.18	0.02
Training Stable	✓	✓	✓	Unstable	Collapsed
Convergence Epoch	42	38	**35**	48	-

As shown in Table 12, smaller α values (0.01, 0.05) maintain stability but converge more slowly and achieve suboptimal performance. Larger values ($\alpha \geq 0.2$) cause coordinate collapse, where the standard deviation of coordinate embeddings approaches zero, indicating all cities converge to similar positions in the latent space.

Temperature Parameter τ for Weight Scaling The temperature parameter τ in Equation 9 scales the MLP outputs before applying tanh, preventing gradient saturation:

Table 13: Effect of temperature τ on gradient flow and performance (TSP-50). †Average gradient norm in coordinate MLP during first 10 epochs. ‡Percentage of tanh outputs with $|w_{ij}| > 0.95$.

Temperature τ	1	5	**10**	20	50
Optimality Gap (%)	0.12	0.04	**0.01**	0.02	0.05
Avg. Gradient Norm†	0.003	0.018	**0.042**	0.038	0.031
Tanh Saturation Rate‡	68%	24%	**8%**	12%	18%
Convergence Epoch	52	40	**35**	36	39

Table 13 shows that $\tau = 10$ achieves the optimal balance. Lower values ($\tau \leq 5$) cause excessive saturation, leading to vanishing gradients and slower convergence. Higher values ($\tau \geq 20$) reduce the non-linearity's effectiveness, diminishing the model's expressiveness.

Joint Impact Analysis We evaluate the joint effect of α and τ on TSP-100 performance:

Table 14: Joint ablation of α and τ on TSP-100 (optimality gap %).

α \ τ	1	5	10	20	50
0.01	0.21	0.15	0.08	0.09	0.12
0.05	0.18	0.09	0.05	0.06	0.08
0.1	0.15	0.06	**0.04**	0.05	0.07
0.2	Unstable	0.18	0.15	0.16	0.19
0.5	Collapsed	Collapsed	Diverged	Diverged	Diverged

The joint ablation confirms that $\alpha = 0.1$ and $\tau = 10$ provide the optimal configuration, with performance degrading smoothly as we deviate from these values.

F.7 Model Efficiency

Although the results in the main text are from the full-scale EDISCO model, it is interesting to see EDISCO's performance under reduced model sizes. Table 15 presents a comprehensive comparison of model efficiency across different architectures and scales, demonstrating how EDISCO's equivariant design enables strong performance even with reduced model capacity and training data.

Table 15: Model efficiency and performance comparison across TSP scales. Training instances shown in thousands (K) with corresponding optimality gaps (%). EDISCO-Full uses 12 EGNN layers with 256 hidden dimension (5.5M parameters), EDISCO-Medium uses 12 layers with 128 hidden dimension (2.6M parameters), and EDISCO-Small uses 8 layers with 128 hidden dimension (1.4M parameters). †12-layer GNN with width 256. *Same architecture as DIFUSCO.

Method	Model Parameters	Training Instances (K) / Optimality Gap (%)				
		TSP-50	TSP-100	TSP-500	TSP-1000	TSP-10000
DIFUSCO†	5.3M	1500 / 0.48	1500 / 1.01	128 / 9.41	64 / 11.24	6.4 / 8.95
DISCO†	5.3M	1500 / 0.16	1500 / 0.58	- / -	- / -	6.4 / 2.90
T2T*	5.3M	1500 / 0.04	1500 / 0.18	128 / 5.09	64 / 8.87	6.4 / 2.92
EDISCO-Full	5.5M	500 / **0.01**	500 / **0.04**	60 / **1.95**	30 / **2.85**	3 / **1.98**
EDISCO-Medium	2.6M	500 / 0.03	500 / 0.08	60 / 2.18	30 / **2.85**	3 / 2.43
EDISCO-Small	1.4M	**300** / 0.08	**300** / 0.7	**40** / 4.18	**20** / 5.21	**2** / 3.18

EDISCO-Full (5.5M parameters) uses 3× less training data than baselines on small instances (500K vs 1.5M) and 2× less on large instances. It achieves 0.01% gap on TSP-50 compared to DIFUSCO's

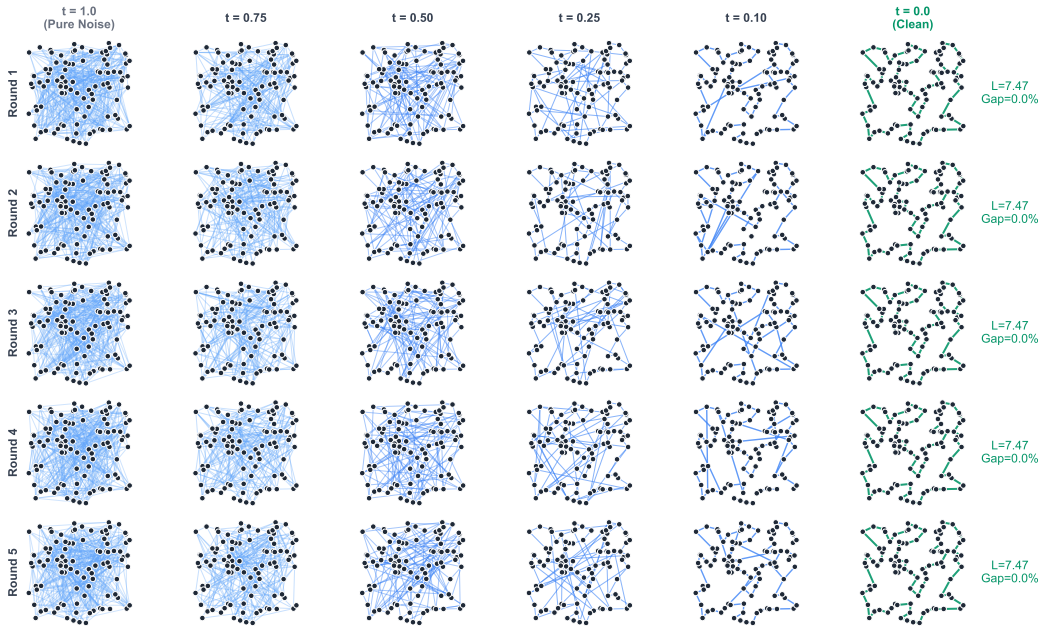

Figure 4: Five rounds of the denoising process of EDISCO on the original TSP instance.

0.48%, and 1.95% on TSP-500 versus DIFUSCO's 9.41%. On TSP-10000, EDISCO-Full achieves 1.98% gap with 3K training instances, outperforming DIFUSCO's 8.95% gap with 6.4K instances.

EDISCO-Medium (2.6M parameters), with less than half the parameters of baseline models, achieves 0.03% gap on TSP-50 and 2.18% on TSP-500. It matches EDISCO-Full's performance on TSP-1000 at 2.85% gap. Compared to T2T, which achieves 0.04% on TSP-50 with 5.3M parameters, EDISCO-Medium achieves comparable performance (0.03%) with 2.6M parameters and the same 500K training instances.

EDISCO-Small (1.4M parameters) uses 3.8× fewer parameters than baselines and requires the least training data: 300K for TSP-50/100, 40K for TSP-500, and 2K for TSP-10000. It achieves 0.08% gap on TSP-50 and 0.7% on TSP-100. On TSP-500, with 40K training instances, it achieves 4.18% gap, compared to DIFUSCO's 9.41% with 128K instances. On TSP-10000, EDISCO-Small achieves 3.18% gap, outperforming DIFUSCO (8.95%) and DISCO (2.90%).

The results for the amount of training data represent the minimum numbers required to ensure the EDISCO converges to the optimal gaps. After this, even with increased training data, there is no noticeable improvement in the optimality gaps.

F.8 Visualization of EDISCO's E(2)-Equivariance

Figure 4 illustrates the denoising process of EDISCO on a standard TSP-100 instance. To empirically validate the E(2) equivariance of our architecture, we present the same denoising process on rotated versions of the instance in Figures 5 and 6. Only the greedy decoder is used without any other post-processing techniques. The visualization shows progression from pure noise ($t = 1.0$) to clean tours ($t = 0.0$) across five independent sampling rounds, demonstrating consistent convergence to high-quality solutions. The similar performance across all three orientations confirms that EDISCO has learned truly rotation-invariant representations.

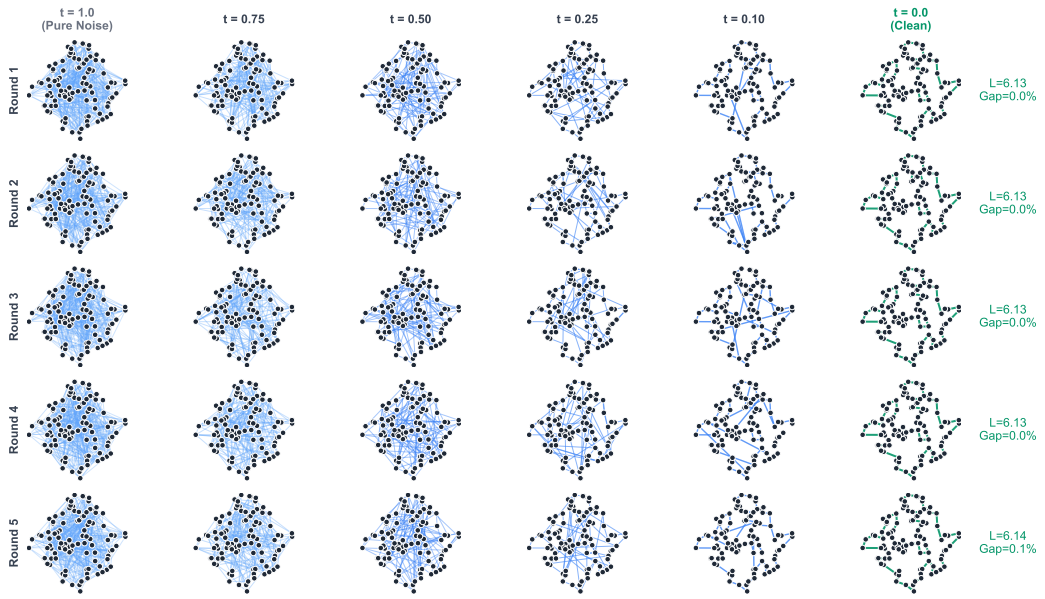

Figure 5: Five rounds of the denoising process of EDISCO on the 45° rotated instance.

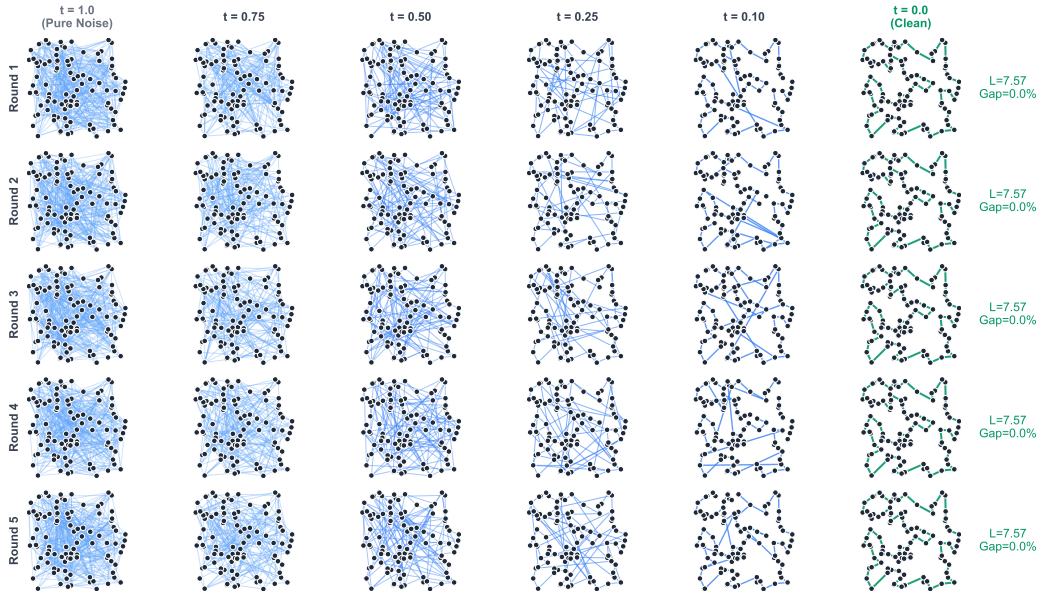

Figure 6: Five rounds of the denoising process of EDISCO on the 90° rotated instance.