EDISCO: Equivariant Continuous-Time Categorical Diffusion for Geometric Combinatorial Optimization

Anonymous Authors[1]

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. It 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 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., 2018; 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, most existing neural network-based approaches 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., 2018; 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., 2021a; Fu et al., 2021). 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., 2024; Bronstein et al., 2021). Such transformations are denoted as E(2). Existing neural network-based approaches mostly fail to capture the geometric structure of GCOPs, which is a major reason for their inefficiency. 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 and generalization 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 dif-

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

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

fusion formulation presents another crucial design decision. While discrete diffusion models have shown promising performances for combinatorial problems (Sun & Yang, 2023; Austin et al., 2021), existing diffusion-based approaches employ discrete-time formulations with certain limitations. Although methods like DDIM (Song et al., 2020a) allow variable step counts at inference, they still rely on fixed discretization schemes that may accumulate approximation errors (Zhang & Chen, 2022; Lu et al., 2022; 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). On the contrary, 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 function evaluations (NFEs) (Song et al., 2020a; Sun et al., 2022).

In this work, we propose EDISCO, which leverages an Equivariant Graph Neural Network (EGNN) architecture that respects geometric symmetries inherent in GCOP 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 and reverse denoising processes, thereby accelerating 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 GCOPs. Preserving geometric 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). Multi-step methods like PNDM (Liu et al., 2022) are adapted via exact posterior sampling (Appendix I.1), achieving 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 achieved 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 (TSPLIB benchmarks, Appendix I.4) and other problems, including Capacitated Vehicle Routing Problem (CVRP, Appendix C), Euclidean Steiner Tree Problem (ESTP, Appendix B), and Maximum Independent Set (MIS, Appendix D). The results validate the generalizability of EDISCO across various domains.

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., 2018; Joshi et al., 2022; Fu et al., 2021). These approaches can be divided into autoregressive and non-autoregressive models.

Autoregressive models (Kool et al., 2018; Kwon et al., 2020) construct solutions sequentially but require extensive training data and struggle with generalization (Joshi et al., 2022). Recent autoregressive methods explore alternative paradigms: COExpander (Ma et al., 2025) introduces adaptive expansion that interpolates between global prediction and local construction, achieving strong performance on TSP and Asymmetric TSP (ATSP) up to 10K nodes; BQ-NCO (Drakulic et al., 2023) reformulates the problem as a Markov Decision Process (MDP) with bisimulation quotienting for improved generalization.

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). The UTSP (Min et al., 2023) demonstrates that unsupervised learning with surrogate losses can achieve competitive TSP performance using only 0.2% of typical training data. 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. Fast-T2T (Li et al., 2024) accelerates diffusion-based methods by optimizing consistency between training and testing, achieving substantial speedups while maintaining competitive solution quality. CADO (Yoon et al., 2024) enhances DIFUSCO by combining RL fine-tuning, but requires high-quality supervised data and expensive RL fine-tuning. Though DIffUCO (Sanokowski et al., 2024) applies unsupervised learning to diffusion models for combinatorial optimization on graph problems (MIS, clique, max-cut), it lacks geometric considerations and does not show results on large-scale Euclidean TSP/CVRP problems.

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., 2024), 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.

2.3. Continuous-Time Diffusion and Sampling Methods

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., 2022) 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., 2020b). 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.

Beyond diffusion, sampling-based approaches offer alternative perspectives on combinatorial optimization. iSCO (Sun et al., 2023) revisits Markov Chain Monte Carlo (MCMC) sampling with expensive 2-opt moves, achieving high-quality solutions on large TSPs but at significantly longer runtimes (tens of GPU-hours). RLSA (Feng & Yang, 2025) employs regularized Langevin dynamics with neural networks to improve exploration on graph CO problems (MIS, clique, max-cut), achieving 80% faster convergence than simulated annealing. Variational approaches have also shown promise: VAG-CO (Sanokowski et al., 2023) learns Boltzmann distributions on Ising-formulated graph optimization tasks (MIS, MVC, MaxCut), while GFlowNets (Zhang et al., 2023) demonstrate strong performance on graph CO through flow-based generative models. These methods target non-geometric graph problems and do not directly address Euclidean GCOPs.

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. This is formulated 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 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 across all problem sizes. Additionally, the continuous-time formulation 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 $\mathbf{X}_0$ is progressively corrupted to noise through a CTMC process. 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 we select $\beta(t) = \beta_{\min} + t(\beta_{\max} - \beta_{\min})$ to be a linear noise schedule with $t \in [0,1]$. We set $\beta_{\min} = 0.1$ and $\beta_{\max} = 1.5$. The selections of the schedule and values are based on comprehensive evaluations shown in Appendix I.5.

The transition probability from time s to t is obtained by solving the Kolmogorov forward equation (Norris, 1998), 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 $\mathbf{X}_0$:

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

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, which is crucial for efficient training. The exponential decay term ensures that as $t \to 1$, the transition probability substantially corrupts the original signal (with our schedule, $P_{ii} \approx 0.60$, $P_{ij} \approx 0.40$) while maintaining mathematical tractability.

Reverse Process The reverse process reconstructs clean data from noise by iteratively applying learned denoising steps. 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. Here, $p_{\text{diffusion}} = q(\mathbf{X}_{t-\Delta t}|\mathbf{X}_t, \hat{\mathbf{X}}_0)$ is the posterior transition probability from Eq. 6 using the predicted $\hat{\mathbf{X}}_0 = s_\theta(\mathbf{X}_t, t, \{\mathbf{c}_i\})$, and $p_{\text{predicted}} = \text{Cat}(\hat{\mathbf{X}}_0)$ is a categorical distribution directly sampling from the network's prediction. This design is grounded in the structure of CTMC reverse dynamics. The exact reverse rate depends on the marginal $p(X_t)$, which must be estimated via the neural network (Campbell et al., 2022). The mixing strategy reflects the uncertainty in this estimation. At large t, predictions are unreliable due to severe corruption. As t decreases, predictions become increasingly reliable. The following proposition formalizes this. All details about the notations are comprehensively explained in the Default Notation.

Proposition 3.1. *For the binary CTMC with forward transition (Eq. 3) and uniform prior:*

(i) The posterior variance is $Var[X_0|X_t] = (1 - e^{-4\bar{\beta}(t)})/4$*, where* $\bar{\beta}(t) = \int_0^t \beta(u)du$.

(ii) For linear noise schedule and small t*:* $Var[X_0|X_t] = \beta_{\min} t + O(t^2)$.

Since the posterior variance scales linearly with t for small t (Proposition 3.1(ii)), the linear mixing weight $w(t) = t$ aligns with this uncertainty profile in the critical low-noise regime. For very small timesteps ($t < 0.1$ or $|\Delta t| < 0.02$), we switch to deterministic argmax transitions. The proof is presented in Appendix E.1, the comprehensive evaluation is shown in Appendix I.6, and the training details are provided in Appendix H.6.

3.3. Equivariant Graph Neural Network Architecture

Geometric Equivariance for GCOP GCOP 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., 2024; 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 $\mathbf{h}_i$ initialized as learned invariant embeddings, edge features $\mathbf{e}_{ij}$ representing pairwise relationships and tour decisions, and coordinates $\mathbf{x}_i$ that evolve during message passing. Figure 1 uses TSP as an example to illustrate that the EGNN architecture 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.

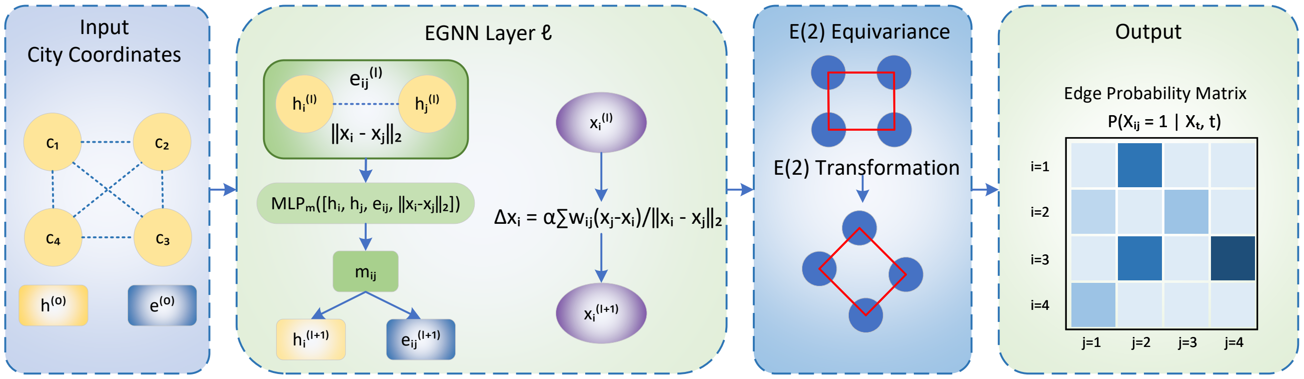

Figure 1. EDISCO's EGNN Architecture Overview. EGNN layers process TSP instances while preserving $E(2)$ equivariance. The network outputs edge probabilities that remain invariant under geometric transformations.

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 τ is selected to be 10, and the step size α is selected to be 0.1. The selections of τ and α are extensively evaluated in Appendix I.7. This specific selection prevents saturation of the tanh function during early training when the MLP outputs may be large. The normalization operation by distance ensures that the update magnitude is independent of the coordinate scale, therefore improving robustness.

Edge features are updated with explicit time conditioning:

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

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:

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

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. The following proposition formalizes this reduction.

Proposition 3.2. *Let $X = \mathbb{R}^{2n}$ denote the space of ordered 2D coordinates for n nodes, and let $G = \text{E}(2)$ be the Euclidean group acting on X by simultaneous rotation, reflection, and translation of all node positions. Let $X_{\text{gp}} \subset X$ be the subset of configurations in general position (no non-trivial stabilizer), and assume $n \geq 3$. Then:*

(i) *The quotient space X_{gp}/G is a smooth manifold of dimension $2n - 3$.*

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

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

Although the dimension reduction from $2n$ to $2n - 3$ appears modest, its impact on learning is substantial. Equivariance reduces the effective hypothesis class by enforcing dependence only on the quotient X_{gp}/G, which can improve sample efficiency under standard generalization bounds (Brehmer et al., 2024). The quotient dimension is $2n - 3$ because E(2) has three continuous degrees of freedom (two for translation, one for rotation). Reflections form a discrete $\mathbb{Z}_2$ subgroup that does not contribute additional continuous dimensions. Notably, for GCOPs, reflected configurations yield identical optimal tours (same edge set), so our architecture naturally handles the full E(2) group. The detailed proof of Proposition 3.2 is given in Appendix E.2. We also prove that the E(2)-Equivariance is preserved during the entire diffusion process in Appendix E.3.

3.4. Inference and Tour Decoding

Solver Selection During inference, the continuous-time formulation allows for a flexible choice of sampling strategies without requiring retraining (Campbell et al., 2022).Multi-step methods (PNDM, DEIS, etc.) are adapted for categorical CTMCs by combining two components: (1) prediction smoothing, which uses Adams-Bashforth coefficients to extrapolate $\hat{p}(\mathbf{X}_0)$ from multiple timesteps, and (2) exact posterior sampling $q(\mathbf{X}_s|\mathbf{X}_t, \hat{p})$ derived from Bayes' rule. This preserves the CTMC structure while accelerating convergence. We extensively evaluate different solvers in Appendix I.2, with methodology detailed in Appendix I.1.

Tour Construction from Edge Probabilities The diffusion model outputs a probability matrix $P \in [0,1]^{n\times n}$ where $P_{ij} = p(X_{ij} = 1)$. Following Sun & Yang (2023), we compute edge scores $s_{ij} = (P_{ij} + P_{ji})/d_{ij}$, where the symmetrization accounts for TSP's undirected nature and d_{ij} is the Euclidean distance. Starting with an empty tour, an edge (i,j) is inserted 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 completing the tour). Cycle detection uses union-find with path compression for near-linear complexity. The 2-opt local search (Lin & Kernighan, 1973) can be optionally applied to further improve the tour.

4. Experiments

We use TSP as the main benchmark for evaluating EDISCO. Extensions to ESTP, CVRP, and MIS are in Appendix B, C, and D.

4.1. Setup

Datasets We follow the standard TSP evaluation protocol from Kool et al. (2018). 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. (2018) 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 (detailed comparison in Appendix I.8), with training configuration reported in Appendix H.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 H.4. Graph sparsification reduces memory complexity from O(n²) to O(kn), enabling training and inference on a single 48GB GPU across all scales from TSP-50 to TSP-10000 (Appendix I.9).

4.2. TSP Results

This section presents the main TSP results from TSP-50 to TSP-1000. Due to space limits, we provide additional results in the appendix: full TSP comparison tables including sampling-based decoding, all baseline methods, and TSP-10000 results (Appendix I.3), TSPLIB benchmark results (Appendix I.4), and comprehensive solver evaluation (Appendix I.2). In this section, we report EDISCO results with two solvers: PNDM (Liu et al., 2022) with 50 steps for best quality, and DEIS-2 (Zhang & Chen, 2022) with 5 steps for fast inference. We report three primary metrics: (1) average tour length, (2) average optimality gap, and (3) total run time. All diffusion-based methods were re-evaluated on identical hardware for fair timing comparison (detailed in Appendix H.2). More experiment details can be found in Appendix H.

Table 1 presents results across TSP-50 to TSP-1000. On small-scale problems, EDISCO-PNDM achieves near-optimal performance with 0.01% gap on TSP-50 and 0.04% on TSP-100, substantially outperforming DIFUSCO (0.48%, 1.01%) and T2T (0.04%, 0.18%). On larger-scale instances, EDISCO-PNDM achieves gaps of 1.95% and 2.85% on TSP-500 and TSP-1000, outperforming DIFUSCO (9.41%, 11.24%), T2T (5.09%, 8.87%), and CADO (2.30%, 3.33%). With greedy+2-opt, EDISCO-PNDM achieves gaps of 0.18% and 0.52% on TSP-500/1000, outperforming CADO (0.24%, 0.69%) while being 3.5× and 2.7× faster. With sampling+2-opt decoding, EDISCO-PNDM achieves state-of-the-art gaps of 0.08% and 0.22%, improving over CADO (0.12%, 0.30%) while being 3.4× and 2.6× faster. The continuous-time formulation enables flexible speed-quality trade-offs: EDISCO-DEIS2 is approximately 9× faster than EDISCO-PNDM while maintaining competitive quality.

EDISCO also scales to TSP-10000 achieving 1.20% gap (Appendix I.3), generalizes to real-world TSPLIB instances with 0.088% average gap (Appendix I.4), and extends to ESTP and CVRP with competitive results (Appendix B, C).

4.3. Problem Size Generalization

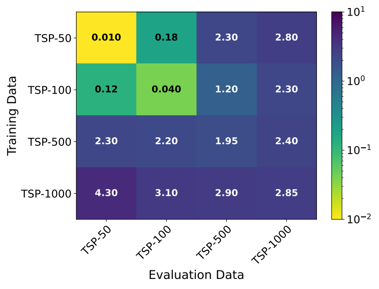

Figure 2. Generalization performance across TSP sizes under greedy decoding.

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.

Table 1. Results on TSP benchmarks from TSP-50 to TSP-1000.† RL: Reinforcement Learning, SL: Supervised Learning, G: Greedy, S: Sampling, 2O: 2-opt. Concorde* is used as the baseline for TSP-50/100; LKH-3 is used as the baseline for TSP-500/1000. Bold: best, underlined: second-best.

Algorithm	Type	TSP-50	TSP-100	TSP-500		TSP-1000	
		Gap	Gap	Gap	Time	Gap	Time
Concorde* (Applegate et al., 2006)	Exact	0.00%	0.00%	–	37.66 m	–	6.65 h
LKH-3 (Helsgaun, 2017)	Heuristic	–	–	0.00%	46.28 m	0.00%	2.57 h
AM (Kool et al., 2018)	RL+G	1.76%	4.53%	20.99%	1.51 m	34.75%	3.18 m
POMO (Kwon et al., 2020)	RL+G	0.64%	1.07%	–	–	–	–
DIFUSCO (Sun & Yang, 2023)	SL+G	0.48%	1.01%	9.41%	5.70 m	11.24%	17.33 m
T2T (Li et al., 2023)	SL+G	0.04%	0.18%	5.09%	4.90 m	8.87%	15.66 m
Fast T2T (Li et al., 2024)	SL+G	0.02%	0.07%	5.94%	0.37 m	6.29%	1.35 m
CADO (Yoon et al., 2024)	SL+RL+G	0.01%	0.08%	2.30%	8.23 m	3.33%	18.42 m
EDISCO-PNDM (ours)	SL+G	**0.01%**	**0.04%**	**1.95%**	2.19 m	**2.85%**	6.84 m
EDISCO-DEIS2 (ours)	SL+G	0.02%	0.06%	2.78%	**0.23 m**	4.42%	**0.75 m**
DIFUSCO (Sun & Yang, 2023)	SL+G+2O	0.09%	0.22%	1.55%	5.75 m	1.86%	17.52 m
T2T (Li et al., 2023)	SL+G+2O	0.02%	0.06%	0.78%	4.98 m	1.25%	15.90 m
Fast T2T (Li et al., 2024)	SL+G+2O	0.01%	0.03%	0.39%	2.17 m	0.58%	8.62 m
CADO (Yoon et al., 2024)	SL+RL+G+2O	0.00%	0.01%	0.24%	8.35 m	0.69%	18.67 m
EDISCO-PNDM (ours)	SL+G+2O	**0.00%**	**0.01%**	**0.18%**	2.35 m	**0.52%**	6.97 m
EDISCO-DEIS2 (ours)	SL+G+2O	0.01%	0.02%	0.26%	**0.40 m**	0.82%	**0.90 m**
DIFUSCO (Sun & Yang, 2023)	SL+S+2O	0.01%	0.02%	0.83%	19.05 m	1.30%	59.53 m
T2T (Li et al., 2023)	SL+S+2O	0.00%	0.00%	0.37%	16.03 m	0.78%	54.67 m
Fast T2T (Li et al., 2024)	SL+S+2O	0.00%	0.00%	0.21%	6.85 m	0.42%	18.28 m
CADO (Yoon et al., 2024)	SL+RL+S+2O	0.00%	0.00%	0.12%	27.01 m	0.30%	61.48 m
EDISCO-PNDM (ours)	SL+S+2O	0.00%	0.00%	**0.08%**	8.03 m	**0.22%**	23.48 m
EDISCO-DEIS2 (ours)	SL+S+2O	0.00%	0.00%	0.12%	**1.00 m**	0.35%	**2.80 m**

†Full comparison with 15+ baselines (Sym-NCO, DIMES, GLOP, DISCO, etc.) and TSP-10000 results in Appendix I.3.

This generalizability outperforms other diffusion methods (Sun & Yang, 2023; Li et al., 2023).

4.4. Cross-Distribution Generalization

To evaluate EDISCO's robustness to distribution shift, we conduct out-of-distribution (OOD) experiments following the protocol from Bi et al. (2022). We evaluate on four distributions: Uniform (in-distribution baseline), Cluster, Explosion, and Implosion. Table 2 presents the results, reporting optimality gap and deterioration metric Det.$(\%) = (\text{Gap}_{\text{OoD}}/\text{Gap}_{\text{Uniform}} - 1) \times 100$. EDISCO achieves an average deterioration of only 4%, significantly outperforming GLOP (15%), DIFUSCO (133%), T2T (687%), and Fast-T2T (1961%). On uniform distribution, EDISCO achieves 0.04% gap, maintaining similarly low gaps on OOD distributions: 0.05% on Cluster, 0.03% on Explosion, and 0.05% on Implosion. The dramatic difference in OOD robustness demonstrates that E(2)-equivariance enables learning geometric invariants rather than distribution-specific patterns, fundamentally improving generalization. The OOD results are visualized in Figure 3, and the empirical visualization confirming rotation-invariant tour generation is provided in Appendix I.10.

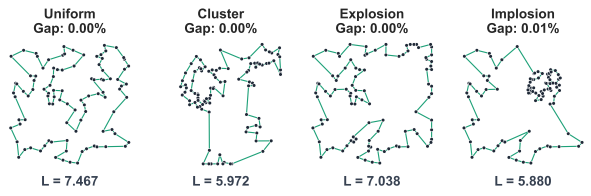

Figure 3. Visualization of EDISCO-generated tours on four standard OOD distributions for TSP-100.

4.5. 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 4 illustrates that EDISCO maintains near-optimal performance even with limited data, achieving gaps below 0.07% with just 10% of training data. Under the same scenario, DIFUSCO achieved an optimality gap of only 2.8% and T2T only 2.1%.

The right panel of Figure 4 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).

Table 2. Cross-distribution generalization on TSP-100 following Bi et al. (2022). All models trained on Uniform distribution only. Det.(%): Deterioration relative to uniform performance. Bold: best, underlined: second-best.

Method	**Uniform**	**Cluster**		**Explosion**		**Implosion**		**Average**	
	Gap↓	Gap↓	Det.↓	Gap↓	Det.↓	Gap↓	Det.↓	Gap↓	Det.↓
AM (Kool et al., 2018)	2.31%	17.97%	678%	3.82%	65%	2.43%	5%	6.63%	249%
AMDKD+EAS (Bi et al., 2022)	0.08%	0.17%	112%	0.05%	-39%	0.08%	1%	0.09%	25%
DIFUSCO (Sun & Yang, 2023)	1.01%	2.87%	184%	1.38%	37%	2.80%	177%	2.02%	133%
T2T (Li et al., 2023)	0.18%	1.50%	733%	0.15%	-17%	2.60%	1344%	1.11%	687%
Fast-T2T (Li et al., 2024)	0.06%	1.18%	1867%	0.03%	**-52%**	2.50%	4067%	0.94%	1961%
GLOP (Ye et al., 2024)	0.09%	0.17%	82%	0.07%	-28%	0.08%	**-10%**	0.10%	15%
EDISCO (ours)	**0.04%**	**0.05%**	**25%**	**0.03%**	-25%	**0.05%**	13%	**0.04%**	**4%**

Table 3. Ablation study on TSP-500 and TSP-1000. The three key components are: (1) EGNN architecture, (2) Continuous-time diffusion, and (3) Adaptive mixing strategy. "w/o X" removes component X while keeping others; "X Only" keeps only component X. Vanilla DIFUSCO uses none of the three. All variants use 50 diffusion steps with PNDM solver (or Euler for discrete-time variants) and greedy decoding. Bold: best, underlined: second-best.

Model Variant	**TSP-500**			**TSP-1000**			**Conv. Epoch**
	Length ↓	Gap% ↓	Time	Length ↓	Gap% ↓	Time	
EDISCO (Full)	**16.87**	**1.95**	2.19 m	**23.78**	**2.85**	**6.84 m**	**35**
w/o Mix Strategy	16.95	2.44	**2.18 m**	23.91	3.41	6.85 m	38
w/o Continuous-Time	17.02	2.86	4.43 m	24.11	4.28	15.58 m	42
w/o EGNN	17.49	5.71	2.31 m	24.85	7.49	7.15 m	51
EGNN Only	17.14	3.58	4.52 m	24.29	5.06	16.12 m	48
Continuous Only	17.72	7.09	2.45 m	25.26	9.27	7.42 m	58
Mix Only	17.61	6.42	5.42 m	25.08	8.52	16.86 m	55
Vanilla DIFUSCO	18.11	9.41	5.70 m	25.72	11.24	17.33 m	61

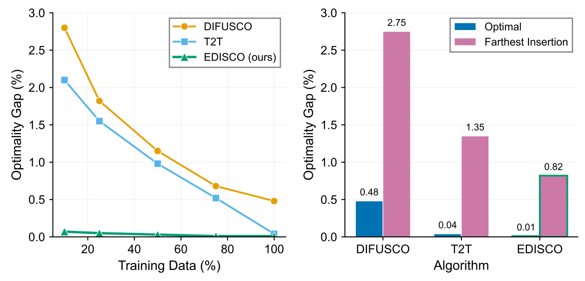

Figure 4. Results on TSP-50 performance. Left: Optimality gap as a function of training set size. Right: Performance comparison when trained on optimal data versus heuristic Farthest Insertion data.

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.

4.6. Ablation Study

Table 3 shows the results of systematically evaluating each component's contribution. EGNN provides the largest individual impact: removing it degrades performance from 1.95% to 5.71% on TSP-500 (2.85% to 7.49% on TSP-1000) and requires 16 additional training epochs. Continuous-time diffusion doubles inference speed (2.19 minutes vs 4.43 minutes on TSP-500) while improving gaps by 0.91% and 1.43%. Adaptive mixing contributes 0.49% and 0.56% improvements. The **EGNN only** case achieves 3.58% and 5.06% gaps, demonstrating 2.63× and 2.22× improvements over vanilla DIFUSCO (9.41% and 11.24%) and isolating EGNN's architectural contribution. Variants without EGNN perform substantially worse (6.42-7.09% gaps), confirming equivariance as the core innovation and contribution.

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 work 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.

References

Ahn, S., Seo, Y., and Shin, J. Learning what to defer for maximum independent sets. In *International conference on machine learning*, pp. 134–144. PMLR, 2020.

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

Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and Van Den Berg, R. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.

Batzner, S., Musaelian, A., Sun, L., Geiger, M., Mailoa, J. P., Kornbluth, M., Molinari, N., Smidt, T. E., and Kozinsky, B. E (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature communications*, 13(1):2453, 2022.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., and Bengio, S. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

Bi, J., Ma, Y., Wang, J., Cao, Z., Chen, J., Sun, Y., and Chee, Y. M. Learning generalizable models for vehicle routing problems via knowledge distillation. *Advances in Neural Information Processing Systems*, 35:31226–31238, 2022.

Böther, M., Kißig, O., Taraz, M., Cohen, S., Seidel, K., and Friedrich, T. What's wrong with deep learning in tree search for combinatorial optimization. *arXiv preprint arXiv:2201.10494*, 2022.

Brehmer, J., Behrends, S., De Haan, P., and Cohen, T. Does equivariance matter at scale? *arXiv preprint arXiv:2410.23179*, 2024.

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

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Butcher, J. C. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.

Campbell, A., Benton, J., De Bortoli, V., Rainforth, T., Deligiannidis, G., and Doucet, A. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022.

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

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

Dagès, T., Weber, S., Lin, Y.-W. E., Talmon, R., Cremers, D., Lindenbaum, M., Bruckstein, A. M., and Kimmel, R. Finsler multi-dimensional scaling: Manifold learning for asymmetric dimensionality reduction and embedding. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 25842–25853, 2025.

Deudon, M., Cournut, P., Lacoste, A., Adulyasak, Y., and Rousseau, L.-M. Learning heuristics for the tsp by policy gradient. In *International conference on the integration of constraint programming, artificial intelligence, and operations research*, pp. 170–181. Springer, 2018.

Dieleman, S., Sartran, L., Roshannai, A., Savinov, N., Ganin, Y., Richemond, P. H., Doucet, A., Strudel, R., Dyer, C., Durkan, C., et al. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022.

Drakulic, D., Michel, S., Mai, F., Sors, A., and Andreoli, J.-M. Bq-nco: Bisimulation quotienting for generalizable neural combinatorial optimization. *arXiv preprint arXiv:2301.03313*, 2023.

Drakulic, D., Michel, S., and Andreoli, J.-M. Goal: A generalist combinatorial optimization agent learner. *arXiv preprint arXiv:2406.15079*, 2024.

Eijkelboom, F., Bartosh, G., Andersson Naesseth, C., Welling, M., and van de Meent, J.-W. Variational flow matching for graph generation. *Advances in Neural Information Processing Systems*, 37:11735–11764, 2024.

Erd6s, P. and Rényi, A. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci*, 5:17–61, 1960.

Esteves, C., Allen-Blanchette, C., Makadia, A., and Daniilidis, K. Learning so (3) equivariant representations with spherical cnns. In *Proceedings of the european conference on computer vision (ECCV)*, pp. 52–68, 2018.

Feng, S. and Yang, Y. Regularized langevin dynamics for combinatorial optimization. *arXiv preprint arXiv:2502.00277*, 2025.

Fu, Z.-H., Qiu, K.-B., and Zha, H. 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.

Gasse, M., Chételat, D., Ferroni, N., Charlin, L., and Lodi, A. Exact combinatorial optimization with graph convolutional neural networks. *Advances in neural information processing systems*, 32, 2019.

Gat, I., Remez, T., Shaul, N., Kreuk, F., Chen, R. T., Synnaeve, G., Adi, Y., and Lipman, Y. Discrete flow matching. *Advances in Neural Information Processing Systems*, 37:133345–133385, 2024.

Goh, Y. L., Cao, Z., Ma, Y., Dong, Y., Dupty, M. H., and Lee, W. S. Hierarchical neural constructive solver for real-world tsp scenarios. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 884–895, 2024.

Graikos, A., Malkin, N., Jojic, N., and Samaras, D. Diffusion models as plug-and-play priors. *Advances in Neural Information Processing Systems*, 35:14715–14728, 2022.

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

Helsgaun, K. An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University*, 12: 966–980, 2017.

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

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

Joshi, C. K., Cappart, Q., Rousseau, L.-M., and Laurent, T. Learning the travelling salesperson problem requires rethinking generalization. *Constraints*, 27(1):70–98, 2022.

Kahng, A. B. and Robins, G. A new class of iterative steiner tree heuristics with good performance. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(7):893–902, 1992.

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35: 26565–26577, 2022.

Kim, M., Park, J., and Park, J. Sym-nco: Leveraging symmetricity for neural combinatorial optimization. *Advances in Neural Information Processing Systems*, 35: 1936–1949, 2022.

Kool, W., Van Hoof, H., and Welling, M. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*, 2018.

Kruskal, J. B. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

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

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

Lamm, S., Sanders, P., Schulz, C., Strash, D., and Werneck, R. F. Finding near-optimal independent sets at scale. In *2016 Proceedings of the eighteenth workshop on algorithm engineering and experiments (ALENEX)*, pp. 138–150. SIAM, 2016.

Lee, J. M. Smooth manifolds. In *Introduction to smooth manifolds*, pp. 1–29. Springer, 2003.

Li, Y., Guo, J., Wang, R., and Yan, J. T2t: From distribution learning in training to gradient search in testing for combinatorial optimization. *Advances in Neural Information Processing Systems*, 36:50020–50040, 2023.

Li, Y., Guo, J., Wang, R., Zha, H., and Yan, J. Fast t2t: Optimization consistency speeds up diffusion-based training-to-testing solving for combinatorial optimization. *Advances in Neural Information Processing Systems*, 37: 30179–30206, 2024.

Li, Z., Chen, Q., and Koltun, V. Combinatorial optimization with graph convolutional networks and guided tree search. *Advances in neural information processing systems*, 31, 2018.

Liao, Z., Chen, J., Wang, D., Zhang, Z., and Wang, J. Bopo: Neural combinatorial optimization via best-anchored and objective-guided preference optimization. *arXiv preprint arXiv:2503.07580*, 2025.

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

Lipman, Y., Chen, R. T., Ben-Hamu, H., Nickel, M., and Le, M. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Lischka, A., Rydin, F., Wu, J., Chehreghani, M. H., and Kulcsár, B. A great architecture for edge-based graph problems like tsp. *arXiv preprint arXiv:2408.16717*, 2024.

Liu, L., Ren, Y., Lin, Z., and Zhao, Z. Pseudo numerical methods for diffusion models on manifolds. *arXiv preprint arXiv:2202.09778*, 2022.

Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems*, 35:5775–5787, 2022.

Luo, F., Lin, X., Liu, F., Zhang, Q., and Wang, Z. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. *Advances in Neural Information Processing Systems*, 36:8845–8864, 2023.

Ma, J., Pan, W., Li, Y., and Yan, J. Coexpander: Adaptive solution expansion for combinatorial optimization. In *Forty-second International Conference on Machine Learning*, 2025.

Ma, Y., Li, J., Cao, Z., Song, W., Guo, H., Gong, Y., and Chee, Y. M. Efficient neural neighborhood search for pickup and delivery problems. *arXiv preprint arXiv:2204.11399*, 2022.

Min, Y., Bai, Y., and Gomes, C. P. Unsupervised learning for solving the travelling salesman problem. *Advances in neural information processing systems*, 36:47264–47278, 2023.

Mirhoseini, A., Goldie, A., Yazgan, M., Jiang, J. W., Songhori, E., Wang, S., Lee, Y.-J., Johnson, E., Pathak, O., Nova, A., et al. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.

Nair, V., Bartunov, S., Gimeno, F., Von Glehn, I., Lichocki, P., Lobov, I., O'Donoghue, B., Sonnerat, N., Tjandraatmadja, C., Wang, P., et al. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*, 2020.

Nazari, M., Oroojlooy, A., Snyder, L., and Takác, M. Reinforcement learning for solving the vehicle routing problem. *Advances in neural information processing systems*, 31, 2018.

Nichol, A. Q. and Dhariwal, P. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PMLR, 2021.

Nordenfors, O., Ohlsson, F., and Flinth, A. Optimization dynamics of equivariant and augmented neural networks. *arXiv preprint arXiv:2303.13458*, 2023.

Norris, J. R. *Markov chains*. Number 2. Cambridge university press, 1998.

Ouyang, W., Wang, Y., Weng, P., and Han, S. Generalization in deep rl for tsp problems via equivariance and local search. *SN Computer Science*, 5(4):369, 2024.

Pan, M., Lin, G., Luo, Y.-W., Zhu, B., Dai, Z., Sun, L., and Yuan, C. Preference optimization for combinatorial optimization problems. *arXiv preprint arXiv:2505.08735*, 2025a.

Pan, W., Xiong, H., Ma, J., Zhao, W., Li, Y., and Yan, J. UniCO: On unified combinatorial optimization via problem reduction to matrix-encoded general TSP. In *The Thirteenth International Conference on Learning Representations*, 2025b.

Pirnay, J. and Grimm, D. G. Self-improvement for neural combinatorial optimization: Sample without replacement, but improvement. *arXiv preprint arXiv:2403.15180*, 2024.

Qin, Y., Madeira, M., Thanou, D., and Frossard, P. Defog: Discrete flow matching for graph generation. *arXiv preprint arXiv:2410.04263*, 2024.

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

Reinelt, G. Tsplib—a traveling salesman problem library. *ORSA journal on computing*, 3(4):376–384, 1991.

Ren, Y., Chen, H., Zhu, Y., Guo, W., Chen, Y., Rotskoff, G. M., Tao, M., and Ying, L. Fast solvers for discrete diffusion models: Theory and applications of high-order algorithms. *arXiv preprint arXiv:2502.00234*, 2025.

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

Sanokowski, S., Berghammer, W., Hochreiter, S., and Lehner, S. Variational annealing on graphs for combinatorial optimization. *Advances in Neural Information Processing Systems*, 36:63907–63930, 2023.

Sanokowski, S., Hochreiter, S., and Lehner, S. A diffusion model framework for unsupervised neural combinatorial optimization. *arXiv preprint arXiv:2406.01661*, 2024.

Särkkä, S. and Solin, A. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.

Satorras, V. G., Hoogeboom, E., and Welling, M. E (n) equivariant graph neural networks. In *International conference on machine learning*, pp. 9323–9332. PMLR, 2021.

Smith-Miles, K., van Hemert, J., and Lim, X. Y. Understanding TSP difficulty by learning from evolved instances. In *Proceedings of the 4th International Conference on Learning and Intelligent Optimization (LION)*, pp. 266–280, Berlin, Heidelberg, 2010. Springer.

Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

Song, Y., Bliek, L., and Zhang, Y. Geometrically invariant and equivariant graph neural networks for tsp algorithm selection and hardness prediction. In *International Conference on Learning and Intelligent Optimization*, pp. 237–252. Springer, 2025.

Sun, H., Yu, L., Dai, B., Schuurmans, D., and Dai, H. Score-based continuous-time discrete diffusion models. *arXiv preprint arXiv:2211.16750*, 2022.

Sun, H., Goshvadi, K., Nova, A., Schuurmans, D., and Dai, H. Revisiting sampling for combinatorial optimization. In *International Conference on Machine Learning*, pp. 32859–32874. PMLR, 2023.

Sun, Z. and Yang, Y. Difusco: Graph-based diffusion solvers for combinatorial optimization. *Advances in neural information processing systems*, 36:3706–3731, 2023.

Thomas, N., Smidt, T., Kearnes, S., Yang, L., Li, L., Kohlhoff, K., and Riley, P. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & operations research*, 40(1): 475–489, 2013.

Vinyals, O., Fortunato, M., and Jaitly, N. Pointer networks. *Advances in neural information processing systems*, 28, 2015.

Wang, H. and Li, P. Unsupervised learning for combinatorial optimization needs meta-learning. *arXiv preprint arXiv:2301.03116*, 2023.

Wang, S., Wang, Y., and Tong, G. Deep-steiner: Learning to solve the euclidean steiner tree problem. In *International Wireless Internet Conference*, pp. 228–242. Springer, 2022.

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

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

Xu, K., Boussemart, F., Hemery, F., and Lecoutre, C. Random constraint satisfaction: Easy generation of hard (satisfiable) instances. *Artificial intelligence*, 171(8-9):514–534, 2007.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Xu, K., Li, J., Zhang, M., Du, S. S., Kawarabayashi, K.-i., and Jegelka, S. What can neural networks reason about? *arXiv preprint arXiv:1905.13211*, 2019.

Ye, H., Wang, J., Liang, H., Cao, Z., Li, Y., and Li, F. 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 38, pp. 20284–20292, 2024.

Yoon, D., Song, H., Lee, K., and Lim, W. Cado: Cost-aware diffusion solvers for combinatorial optimization through rl fine-tuning. In *ICML 2024 Workshop on Structured Probabilistic Inference $\{\backslash\&\}$ Generative Modeling*, 2024.

Zachariasen, M. T. Optimal interconnection trees in the plane: Theory, algorithms and applications. *Networks*, 33 (2):125–143, 1999.

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

Zhang, D., Dai, H., Malkin, N., Courville, A. C., Bengio, Y., and Pan, L. Let the flows tell: Solving graph combinatorial problems with gflownets. *Advances in neural information processing systems*, 36:11952–11969, 2023.

Zhang, Q. and Chen, Y. Fast sampling of diffusion models with exponential integrator. *arXiv preprint arXiv:2204.13902*, 2022.

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

Zhao, H., Yu, K., Huang, Y., Yi, R., Zhu, C., and Xu, K. Disco: Efficient diffusion solver for large-scale combinatorial optimization problems. *arXiv preprint arXiv:2406.19705*, 2024.

Zheng, K., Lu, C., Chen, J., and Zhu, J. Dpm-solver-v3: Improved diffusion ode solver with empirical model statistics. *Advances in Neural Information Processing Systems*, 36:55502–55542, 2023.

Zheng, Z., Zhou, C., Xialiang, T., Yuan, M., and Wang, Z. Udc: A unified neural divide-and-conquer framework for large-scale combinatorial optimization problems. *Advances in Neural Information Processing Systems*, 37: 6081–6125, 2024.

Appendices

Default Notation . . . 15
A. Scope and Limitations . . . 16
B. Extension to Euclidean Steiner Tree Problem . . . 17
C. Extension to Capacitated Vehicle Routing Problem . . . 18
D. Extension to Maximum Independent Set . . . 20
E. Proofs . . . 22
- E.1 Proof of Proposition 3.1 . . . 22
- E.2 Proof of Proposition 3.2 . . . 22
- E.3 Proof of E(2)-Equivariance in EDISCO . . . 24

F. Extended Related Work . . . 25
- F.1 Foundational Neural Combinatorial Optimization . . . 25
- F.2 Alternative Architectures and Scaling Approaches . . . 25
- F.3 Discrete Diffusion Foundations and Variants . . . 25
- F.4 Theoretical Foundations and Sample Complexity . . . 25
- F.5 Hybrid and Practical Approaches . . . 26

G. Architecture Details . . . 26
- G.1 Network Architecture Overview . . . 26
- G.2 Feature Representations and Initialization . . . 26
- G.3 Equivariant Message Passing Mechanism . . . 26
- G.4 Continuous-Time Diffusion Specifications . . . 27

H. Additional Experiment Details . . . 27
- H.1 Performance Metrics . . . 27
- H.2 Hardware Platform . . . 27
- H.3 Randomness in Results . . . 28
- H.4 Data Generation Process . . . 28
- H.5 Model Architecture Specifications . . . 28
- H.6 Training Configuration . . . 28

I. Additional Results . . . 29
- I.1 Multi-Step Methods for Categorical CTMC . . . 29
- I.2 Solver Evaluation on TSP-500 . . . 29
- I.3 Full TSP Results . . . 31
- I.4 TSPLIB Results . . . 33
- I.5 Noise Schedule Design for TSP Diffusion . . . 34
- I.6 Evaluation on Adaptive Mixing Parameters . . . 35
- I.7 Evaluation on Architectural Hyperparameters . . . 36
- I.8 Model Efficiency . . . 38
- I.9 Training Resource Requirements . . . 38
- I.10 Visualization of EDISCO's E(2)-Equivariance . . . 39

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
$\mathbf{x}_i^{(\ell)}$	Coordinates for node i at layer ℓ (evolve equivariantly)
$\mathbf{m}_{ij}^{(\ell)}$	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: $\mathbb{R}^2 \rtimes O(2)$ (translations, rotations, reflections)
$O(2)$	Orthogonal group (rotations and reflections)
$SO(2)$	Special orthogonal group (rotations only)
$g \in E(2)$	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
Cat$(\cdot)$	Categorical distribution
δ_{ij}	Kronecker delta (1 if $i = j$, 0 otherwise)
$\mathcal{L}$	Loss function
NFE	Number of function evaluations
$O(t^k)$	Big-O asymptotic order: $f(t) = O(t^k)$ means $\|f(t)\| \leq Ct^k$ for sufficiently small t
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
MLP$(\cdot)$	Multi-layer perceptron
LayerNorm$(\cdot)$	Layer normalization
SiLU$(\cdot)$	Sigmoid Linear Unit activation
softmax$(\cdot)$	Softmax function
argmax$(\cdot)$	Argument of the maximum

A. Scope and Limitations

EDISCO is specifically designed for geometric combinatorial optimization problems where solutions exhibit E(2) symmetries. This design choice provides substantial benefits for problems like TSP, CVRP, and ESTP. For problems such as Maximum Independent Set (MIS) and Max-Cut on arbitrary graphs, node positions do not carry semantic meaning, but only the graph topology matters. Without geometric structure, E(2)-equivariance provides no inductive bias. However, the underlying continuous-time categorical diffusion framework remains applicable to such non-geometric problems when paired with standard GNN architectures instead of EGNN. We validate this in Appendix D, where we apply our CTMC-based diffusion framework with a standard GNN encoder to MIS benchmarks. The results demonstrate competitive performance against existing methods, confirming that while E(2)-equivariance is specific to geometric problems, the continuous-time diffusion methodology generalizes effectively to graph optimization tasks without geometric structure.

Similarly, EDISCO, in its current geometric EGNN formulation, does not support the ATSP, as costs are provided only in a cost matrix, without explicit node coordinates. The distance from city i to city j differs from j to i (i.e., $d_{ij} \neq d_{ji}$). This limitation is architectural. EDISCO's E(2)-equivariant EGNN fundamentally requires Euclidean coordinates to preserve geometric symmetries, and any distance induced by Euclidean coordinates is necessarily symmetric. Therefore, EDISCO cannot directly represent the arbitrary asymmetric cost matrices required for ATSP.

But this is not an EDISCO-specific limitation compared to other neural TSP solvers. Coordinate-based attention methods (AM (Kool et al., 2018), POMO (Kwon et al., 2020), Pointer Networks (Vinyals et al., 2015)) are designed with positional encodings for Euclidean TSP. Although graph-based diffusion methods (DIFUSCO (Sun & Yang, 2023), T2T (Li et al., 2023), Fast-T2T (Li et al., 2024)) use GNNs that could theoretically process cost matrices only, they have only still been designed and evaluated on coordinate-based Euclidean TSP. EDISCO's E(2)-equivariance provides a stronger architectural constraint that fundamentally requires coordinates.

Current ATSP solvers incorporate directional edge information, such as full cost-matrix encoding (MatNet (Kwon et al., 2021)) or dual incoming/outgoing attention mechanisms (GREAT (Lischka et al., 2024)). These architectural features are incompatible with E(2)-equivariant coordinate encoders.

Promising Future Directions: However, the continuous-time diffusion framework of EDISCO is generalizable. We identify two complementary research directions: (1) **Replace the encoder**: Adopt ATSP-capable encoders (e.g., MatNet, GREAT) while retaining the continuous-time diffusion framework. (2) **Learn coordinate embeddings**: Transform asymmetric cost matrices into approximate coordinate representations using techniques such as Finsler Multi-Dimensional Scaling (Dagès et al., 2025), which extends classical MDS to handle asymmetric dissimilarities by embedding into Finsler spaces rather than symmetric Riemannian manifolds, or neural metric learning approaches that learn distance-preserving coordinate embeddings. Such learned coordinates could enable approximate E(2)-equivariance for ATSP instances that admit near-geometric structure.

It is worth noting that the specialization to coordinate-based geometric problems is a deliberate architectural choice that leverages the strong inductive bias from E(2)-equivariance for problems with inherent geometric structure. By constraining the model to respect E(2) symmetries, EDISCO achieves superior sample efficiency (requiring only 33-50% of training data compared to baselines) and better generalization across problem sizes and distributions. E(2)-equivariance provides substantial advantages when problems possess inherent geometric structure and symmetries. Applying it to problems lacking these properties would simply discard these advantages without additional benefit.

B. Extension to Euclidean Steiner Tree Problem

Problem Formulation and Equivariance Preservation The Euclidean Steiner Tree Problem (ESTP) (Zachariasen, 1999) seeks to find a minimum-length tree connecting a given set of terminal points in Euclidean space, with the option to introduce additional Steiner points to reduce total tree length. Unlike TSP which forms cycles, ESTP produces acyclic tree structures, making it a natural testbed for demonstrating EDISCO's applicability beyond routing problems.

To preserve E(2) equivariance in ESTP, we separate equivariant and invariant features:

- **Equivariant features**: Terminal and candidate Steiner point coordinates $\mathbf{c} \in \mathbb{R}^{n\times 2}$ that transform under rotations and translations
- **Invariant features**: Terminal indicator $\mathbb{1}_{\text{terminal}}$ that remains unchanged under geometric transformations

The optimal Steiner tree for a rotated/translated/reflected point set is simply the rotated/translated version of the original optimal tree. This geometric property makes E(2)-equivariance theoretically grounded: minimizing $\sum_{\text{edges}} \|e_i\|_2$ is invariant under Euclidean transformations since edge lengths are preserved: $\|(R\mathbf{x}_i+\mathbf{t})-(R\mathbf{x}_j+\mathbf{t})\|_2 = \|R(\mathbf{x}_i-\mathbf{x}_j)\|_2 = \|\mathbf{x}_i-\mathbf{x}_j\|_2$ for rotation R and translation $\mathbf{t}$.

Feature Separation in EGNN Layers Similar to TSP, ESTP initializes node embeddings from invariant features:

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

where $\mathbb{1}_{\text{terminal}}(i)$ indicates whether node i is a required terminal point or a candidate Steiner point. This ensures that geometric transformations of coordinates do not affect the initial node representations, maintaining strict equivariance.

Tree-Aware Greedy Decoding The greedy decoder for ESTP constructs a minimum spanning tree connecting selected nodes while ensuring all terminals are included.

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 tree structures.

The tree is constructed using a greedy approach that ensures connectivity while selecting high-scoring edges. Starting with all terminal points as required nodes $\mathcal{T}$ and candidate Steiner points $\mathcal{S}$, we iteratively add edges to build a connected tree. At each step, we select the highest-scoring edge $e^* = (i,j) = \arg\max_{(i,j)\in\mathcal{E}} s_{ij}$ that connects two components without creating cycles, similar to Kruskal's algorithm (Kruskal, 1956) but weighted by learned edge scores rather than pure distances. Candidate Steiner points are only included if they improve the overall tree structure by serving as connection hubs.

To ensure solution completeness, all terminal points must be connected in the final tree. Any disconnected terminals are connected to the main tree using the shortest available edges. This guarantees feasibility while maintaining the geometric structure learned by the equivariant network.

We evaluate EDISCO on Steiner Tree benchmarks with 10, 20, and 50 terminals, following the evaluation protocols in Wang et al. (2022). Each instance includes an equal number of candidate Steiner points uniformly sampled from the unit square $[0, 1]^2$.

Table 4. Results on Steiner-10, Steiner-20, and Steiner-50. RL: Reinforcement Learning, SL: Supervised Learning, G: Greedy. GeoSteiner* represents the baseline for computing the gap (exact solver). MST (Minimum Spanning Tree) provides a simple upper bound by connecting all terminals without Steiner points. Adding optimal Steiner points can only reduce total tree length. Steiner Insertion (SI) is a classical heuristic. Deep-Steiner is the first RL-based method. All neural methods trained on same dataset of 10,000 instances per size.

Algorithm	**Type**	**Steiner-10**			**Steiner-20**			**Steiner-50**		
		Length↓	Gap↓	Time↓	Length↓	Gap↓	Time↓	Length↓	Gap↓	Time↓
GeoSteiner* (Zachariasen, 1999)	Exact	2.62	0.00%	24 s	3.63	0.00%	2.8 m	5.78	0.00%	18.5 m
MST (Upper Bound)	Heuristic	2.84	8.40%	<1 s	3.89	7.16%	<1 s	6.15	6.40%	<1 s
Steiner Insertion (Kahng & Robins, 1992)	Heuristic	2.71	3.44%	<1 s	3.76	3.58%	<1 s	5.97	3.29%	1 s
Learning-Based Greedy Decoding										
Deep-Steiner (Wang et al., 2022)	RL+G	2.73	4.20%	2 s	3.81	4.96%	3 s	6.08	5.19%	7 s
DIFUSCO (Sun & Yang, 2023)	SL+G	2.77	5.73%	2 s	3.81	4.96%	3 s	6.05	4.67%	7 s
T2T (Li et al., 2023)	SL+G	2.73	4.20%	2 s	3.76	3.58%	3 s	5.97	3.29%	8 s
FastT2T (Li et al., 2024)	SL+G	2.68	2.29%	1 s	3.71	2.20%	2 s	5.91	2.25%	4 s
EDISCO with 50-step PNDM (ours)	SL+G	**2.66**	**1.53%**	2 s	**3.68**	**1.38%**	3 s	**5.87**	**1.56%**	8 s
EDISCO with 5-step DEIS-2 (ours)	SL+G	2.69	2.67%	**0.15 s**	3.72	2.48%	**0.35 s**	5.93	2.60%	**0.7 s**

Analysis of Results Table 4 demonstrates EDISCO's performance on the Euclidean Steiner Tree Problem across different instance sizes, comparing against both classical heuristics and learning-based methods. We present two configurations offering distinct quality-speed trade-offs:

With 50-step PNDM and greedy decoding, EDISCO achieves consistent $\sim$1.5% optimality gaps across all problem sizes (1.53% on Steiner-10, 1.38% on Steiner-20, 1.56% on Steiner-50), substantially outperforming the classical Steiner Insertion heuristic (3.44%, 3.58%, 3.29% respectively) by more than 2×.

We retrained the three competing diffusion-based baselines on the same hardware setting for ESTP as a comparison: DIFUSCO (5.73%, 4.96%, 4.67%), T2T (4.20%, 3.58%, 3.29%), and FastT2T (2.29%, 2.20%, 2.25%). EDISCO achieves 1.5-3.7× better optimality gaps, demonstrating that E(2)-equivariance provides crucial inductive bias beyond standard diffusion frameworks. EDISCO also outperforms Deep-Steiner (4.20%, 4.96%, 5.19%), the first RL-based approach, by 2.7-3.6×.

The 5-step DEIS-2 configuration provides ultra-fast inference (0.15s, 0.35s, 0.7s) while achieving competitive quality (2.67% on Steiner-10, 2.48% on Steiner-20, 2.60% on Steiner-50)—approximately 9-13× faster than PNDM-50 while remaining well below the classical Steiner Insertion baseline (3.44%, 3.58%, 3.29%). This configuration significantly outperforms the MST upper bound (8.40%, 7.16%, 6.40%) and all learning-based methods, making it ideal for real-time applications.

C. Extension to Capacitated Vehicle Routing Problem

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 the equivariant and invariant 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

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 Similar to TSP, CVRP initializes node embeddings from invariant features:

$$h_i^{(0)} = \text{NodeEmbed}([d_i, \mathbb{1}_{\text{depot}}(i)]) \tag{13}$$

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. (2018). The vehicle capacity is 30 for CVRP-20, 40 for CVRP-50, and 50 for CVRP-100. The customer demands are uniformly sampled from $\{1, ..., 9\}$.

Table 5. Results on CVRP-20, CVRP-50, and CVRP-100. RL: Reinforcement Learning, SL: Supervised Learning, G: Greedy, S: Sampling. **HGS results** are from Vidal et al. (2013). LKH-3 results serve as reference baseline. POMO and Sym-NCO results are from their respective papers.

Algorithm	**Type**	**CVRP-20**			**CVRP-50**			**CVRP-100**		
		Cost↓	Gap↓	Time↓	Cost↓	Gap↓	Time↓	Cost↓	Gap↓	Time↓
Gurobi (Gurobi Optimization, LLC, 2020)	Exact	6.10	0.00%	–	–	–	–	–	–	–
HGS (Vidal et al., 2013)	**Heuristic**	**6.10**	**0.00%**	**1 h**	**10.38**	**0.00%**	**3 h**	**15.65**	**0.00%**	**5 h**
LKH-3 (Helsgaun, 2017)	Heuristic	6.14	0.58%	2 h	10.38	0.00%	7 h	15.65	0.00%	12 h
Neural Methods - Greedy Decoding										
AM (Kool et al., 2018)	RL+G	6.40	4.97%	<1 s	10.98	5.86%	1 s	16.80	7.34%	3 s
POMO (Kwon et al., 2020)	RL+G	6.35	3.72%	<1 s	10.74	3.52%	1 s	16.13	3.09%	3 s
Sym-NCO (Kim et al., 2022)	RL+G	–	–	–	–	–	–	**16.10**	**2.88%**	3 s
EDISCO with 50-step PNDM50 (ours)	SL+G	**6.21**	**1.41%**	1 s	**10.63**	**2.46%**	2 s	16.15	3.17%	5 s
EDISCO-DEIS2 (ours)	SL+G	6.23	2.01%	**0.1 s**	10.66	3.51%	**0.2 s**	16.22	4.53%	**0.5 s**
Neural Methods - Sampling/Augmentation										
AM (Kool et al., 2018)	RL+S (1280)	6.25	2.49%	3 m	10.62	2.40%	7 m	16.23	3.72%	30 m
POMO (no aug) (Kwon et al., 2020)	RL+G	6.17	0.82%	1 s	10.49	1.14%	4 s	15.83	1.13%	19 s
POMO (×8 aug) (Kwon et al., 2020)	RL+G	**6.14**	**0.21%**	5 s	10.42	0.45%	26 s	15.73	0.51%	2 m
Sym-NCO (Kim et al., 2022)	RL+S (100)	–	–	–	–	–	–	15.87	1.40%	16 s
EDISCO with 50-step PNDM50 (ours)	SL+S (16)	6.15	0.33%	4 s	**10.41**	**0.35%**	7 s	**15.71**	**0.38%**	18 s

Analysis of Results Table 5 demonstrates EDISCO's performance on small-medium scale CVRP (20-100 customers). We present two solver configurations offering distinct quality-speed trade-offs:

Greedy decoding (PNDM-50): EDISCO achieves competitive results with 1.41% gap on CVRP-20 and 2.46% on CVRP-50, outperforming AM (4.97%, 5.86%) and POMO (3.72%, 3.52%). On CVRP-100, EDISCO achieves 3.17% gap, slightly trailing Sym-NCO (2.88%) with comparable inference time (5s vs 3s). The 5-step DEIS-2 variant provides 10× speedup

(0.5s on CVRP-100) while maintaining reasonable quality (4.53% gap), demonstrating the flexibility of continuous-time diffusion for adjusting quality-speed trade-offs without retraining.

Sampling (16 samples): With multiple samples, EDISCO achieves best neural solver results on CVRP-50 and CVRP-100 (0.35% and 0.38% gaps), though trailing POMO with 8× augmentation on CVRP-20 (0.33% vs 0.21%). The runtime remains competitive (18s vs 16s for Sym-NCO on CVRP-100).

Scalability Our CVRP experiments focus on **small-medium scale (20-100 customers)** as proof-of-concept for architectural extensibility to constrained routing problem. This demonstrates that E(2)-equivariance benefits transfer beyond symmetric TSP to problems with capacity constraints and depot requirements.

Large-scale CVRP (500-1000+ customers) represents a fundamentally different challenge requiring specialized approaches, such as GLOP (Ye et al., 2024) and UDC (Zheng et al., 2024). These methods universally adopt a two-stage **divide-and-conquer strategy** that exploits the insight that CVRP can be decomposed into multiple sub-TSP problems. In contrast, competingdiffusion-based methods (DIFUSCO (Sun & Yang, 2023), T2T (Li et al., 2023), Fast-T2T (Ma et al., 2022)) have **no reported CVRP experiments**, highlighting the difficulty of scaling single end-to-end diffusion models to large-scale constrained routing.

Future direction Scaling EDISCO to large-scale CVRP is a valuable future work. A promising direction is leveraging E(2)-equivariance to develop novel partition methods. The geometric equivariance properties that enable EDISCO to learn rotation and translation invariant solution patterns could be particularly valuable for the clustering and decomposition phase, combining EDISCO's strengths with partition-based strategies.

D. Extension to Maximum Independent Set

Problem Formulation and Motivation The Maximum Independent Set (MIS) problem is a fundamental graph optimization problem. Given an undirected graph $G = (V, E)$, an independent set is a subset of vertices $S \subseteq V$ such that no two vertices in S are adjacent in G. MIS seeks to find an independent set of maximum cardinality. Unlike TSP, CVRP, and ESTP, MIS is defined purely on graph topology without geometric coordinates, making it a non-geometric combinatorial optimization problem where E(2)-equivariance provides no inductive bias.

We include MIS experiments to demonstrate the generality of EDISCO's continuous-time categorical diffusion framework beyond geometric problems. Since competing diffusion methods (DIFUSCO, T2T, Fast T2T) all report MIS results, this extension enables direct comparison of the underlying diffusion formulations when geometric structure is absent.

Architectural Adaptation for Non-Geometric Problems As discussed in Section A, EDISCO's E(2)-equivariance provides no benefit for problems like MIS where node positions carry no semantic meaning. To evaluate the continuous-time categorical diffusion framework in isolation, we replace the E(2)-equivariant EGNN with a standard message-passing GNN that maintains the same depth (12 layers) and hidden dimensions (64/256) but omits coordinate processing and equivariant updates. The prediction target changes from edge probabilities to node inclusion probabilities $p(v_i \in S)$ for each vertex, while the continuous-time categorical diffusion framework, including the CTMC formulation, adaptive mixing strategy, and multi-step solvers, remains unchanged.

This configuration isolates the contribution of continuous-time diffusion from geometric equivariance, enabling us to assess whether EDISCO's diffusion framework offers benefits beyond geometric inductive bias.

Datasets and Evaluation Protocol Following the standard evaluation protocol from prior work (Sun & Yang, 2023; Li et al., 2023; 2024), we evaluate on two benchmark datasets. The first consists of RB graphs (Xu et al., 2007), random benchmark graphs with 200-300 vertices known for their challenging structure, using 90,000 training instances and 500 test instances. The second consists of Erdős-Rényi (ER) graphs (Erd6s & Rényi, 1960), randomly generated graphs with 700-800 nodes and edge probability $p = 0.15$, using 163,840 training instances and the standard test set from Qiu et al. (2022). Reference solutions for both datasets are computed using KaMIS (Lamm et al., 2016), a state-of-the-art heuristic solver.

Greedy Decoding for MIS The diffusion model outputs node probabilities $P \in [0, 1]^n$ where $P_i = p(v_i \in S)$. We construct feasible independent sets through greedy decoding by first sorting nodes by predicted probability in descending order, then iteratively adding nodes to the solution provided they do not violate independence (i.e., no neighbor is already selected), and continuing until no more nodes can be added. This procedure mirrors the greedy tour construction used for TSP, adapted for the node-selection formulation of MIS.

Table 6. Results on Maximum Independent Set benchmarks. RB-[200-300]: Random Benchmark graphs with 200-300 nodes. ER-[700-800]: Erdős-Rényi graphs with 700-800 nodes and $p = 0.15$. TS: Tree Search, UL: Unsupervised Learning, G: Greedy, S: Sampling. KaMIS serves as the reference baseline. Results for baselines are from Li et al. (2024).

Algorithm	**Type**	**RB-[200-300]**			**ER-[700-800]**		
		Size↑	Drop↓	Time↓	Size↑	Drop↓	Time↓
KaMIS (Lamm et al., 2016)	Heuristic	20.10	–	1h24m	44.87	–	52.13m
Gurobi (Gurobi Optimization, LLC, 2020)	Exact	19.98	0.01%	47m34s	41.28	7.78%	50.00m
Learning-Based Methods with Greedy Decoding							
Intel (Li et al., 2018)	SL+G	–	–	–	34.86	22.31%	6.06m
DIMES (Qiu et al., 2022)	RL+G	–	–	–	38.24	14.78%	6.12m
DIFUSCO (T_s=100) (Sun & Yang, 2023)	SL+G	18.52	7.81%	16m3s	37.03	18.53%	5m30s
T2T (T_s=50, T_g=30) (Li et al., 2023)	SL+G	18.98	5.49%	20m58s	39.81	11.28%	7m7s
Fast T2T (T_s=1, T_g=1) (Li et al., 2024)	SL+G	19.37	3.51%	1m18s	40.25	10.30%	25s
Fast T2T (T_s=5, T_g=5) (Li et al., 2024)	SL+G	**19.49**	**2.89%**	4m44s	**40.68**	**9.34%**	1m32s
EDISCO-PNDM (ours)	SL+G	19.12	4.78%	3m25s	40.05	10.74%	1m48s
EDISCO-DEIS2 (ours)	SL+G	18.85	6.14%	**21s**	39.42	12.14%	**11s**
Learning-Based Methods with Sampling/Search Decoding							
Intel (Li et al., 2018)	SL+TS	18.47	8.11%	13m4s	38.80	13.43%	20.00m
DGL (Böther et al., 2022)	SL+TS	17.36	13.61%	12m47s	37.26	16.96%	22.71m
LwD (Ahn et al., 2020)	RL+S	–	–	–	41.17	8.25%	6.33m
GFlowNets (Zhang et al., 2023)	UL+S	19.18	4.57%	32s	41.14	8.53%	2.92m
DIFUSCO (T_s=100) (Sun & Yang, 2023)	SL+S	19.13	4.79%	20m28s	39.12	12.81%	21m43s
T2T (T_s=50, T_g=30) (Li et al., 2023)	SL+S	19.38	3.53%	30m18s	41.41	7.72%	27m45s
Fast T2T (T_s=1, T_g=1) (Li et al., 2024)	SL+S	19.53	2.74%	1m59s	40.98	8.66%	1m19s
Fast T2T (T_s=5, T_g=5) (Li et al., 2024)	SL+S	**19.70**	**1.90%**	6m59s	**41.73**	**6.99%**	5m51s
EDISCO-PNDM (ours)	SL+S	19.45	3.13%	5m42s	41.22	8.13%	4m35s
EDISCO-DEIS2 (ours)	SL+S	19.18	4.48%	**30s**	40.65	9.40%	**28s**

Analysis of Results Table 6 presents EDISCO's performance on MIS benchmarks, comparing against heuristic solvers, exact solvers, and state-of-the-art learning-based methods including diffusion approaches. With greedy decoding, EDISCO-PNDM achieves 4.78% drop on RB graphs and 10.74% on ER graphs, outperforming DIFUSCO (7.81%, 18.53%) and performing comparably to T2T (5.49%, 11.28%). With sampling decoding, EDISCO-PNDM achieves 3.13% and 8.13% drops, again outperforming DIFUSCO (4.79%, 12.81%) while remaining competitive with T2T (3.53%, 7.72%).

Fast T2T (Li et al., 2024) achieves the best neural solver results through its optimization consistency training, which directly learns mappings from noise to optimal solutions. EDISCO trails Fast T2T (T_s=5, T_g=5) by approximately 1-2% in drop rate. This gap is expected since Fast T2T's consistency-based approach is specifically designed for rapid, high-quality generation, while EDISCO's continuous-time formulation prioritizes flexibility and geometric awareness—properties that provide no benefit for MIS but are central to EDISCO's advantages on geometric problems.

Despite the absence of geometric structure, EDISCO-DEIS2 provides the fastest inference among all diffusion methods (21s greedy, 30s sampling on RB graphs), approximately 4× faster than Fast T2T while maintaining reasonable quality. This demonstrates that EDISCO's continuous-time framework enables flexible solver selection for real-time applications even on problems where geometric equivariance is irrelevant.

The results confirm that EDISCO's continuous-time categorical diffusion framework provides competitive performance on non-geometric problems, though without the substantial improvements observed on geometric problems (TSP, CVRP, ESTP) where E(2)-equivariance provides strong inductive bias. This validates our architectural design philosophy: the continuous-time diffusion formulation is broadly applicable across problem domains, while the geometric benefits from E(2)-equivariance are realized specifically when problems possess inherent Euclidean structure.

E. Proofs

E.1. Proof of Proposition 3.1

Proof. The adaptive mixing strategy is grounded in CTMC theory. The exact reverse rate from state i to j follows from the time-reversal formula for CTMCs (Campbell et al., 2022):

$$\bar{Q}_{ij}(t) = \frac{p(X_t = j)}{p(X_t = i)} Q_{ji}(t)$$

Since the marginal $p(X_t)$ is intractable, we estimate it via the neural network's prediction of $p(X_0|X_t)$.

Part (i): Let $\gamma(t) = e^{-2\bar{\beta}(t)}$. From the forward transition (Eq. 3):

$$P(X_t = 1|X_0 = 1) = \frac{1+\gamma}{2}, \quad P(X_t = 1|X_0 = 0) = \frac{1-\gamma}{2}$$

With uniform prior $P(X_0 = 1) = 1/2$, the marginal is $P(X_t = 1) = 1/2$. By Bayes' rule:

$$P(X_0 = 1|X_t = x) = \frac{1 + (2x-1)\gamma(t)}{2}$$

For a Bernoulli with parameter p, variance is $p(1-p)$. Thus:

$$\text{Var}[X_0|X_t] = \frac{1+(2x-1)\gamma}{2} \cdot \frac{1-(2x-1)\gamma}{2} = \frac{1-\gamma^2}{4} = \frac{1-e^{-4\bar{\beta}(t)}}{4}$$

Part (ii): For small $\bar{\beta}(t)$, Taylor expansion gives $e^{-4\bar{\beta}(t)} \approx 1 - 4\bar{\beta}(t)$. With linear schedule $\beta(u) = \beta_{\min} + u(\beta_{\max} - \beta_{\min})$:

$$\bar{\beta}(t) = \beta_{\min} t + \frac{\beta_{\max} - \beta_{\min}}{2} t^2 = \beta_{\min} t + O(t^2)$$

Therefore: $\text{Var}[X_0|X_t] \approx \beta_{\min} t + O(t^2)$. □

E.2. Proof of Proposition 3.2

Proof. We establish each claim systematically.

Part (i): The Euclidean group $\mathrm{E}(2) = \mathbb{R}^2 \rtimes \mathrm{O}(2)$ consists of translations ($\mathbb{R}^2$) and orthogonal transformations ($\mathrm{O}(2)$, which includes both rotations and reflections). As a Lie group, $\mathrm{E}(2)$ has dimension 3: two parameters for translation and one for rotation angle. Reflections form a discrete $\mathbb{Z}_2$ component that does not contribute to the continuous dimension. The action of $g = (t, R) \in \mathrm{E}(2)$ on a configuration $\mathbf{x} = (x'_1, \ldots, x'_n) \in X$ is given by:

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

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

To show the action is free, assume $n \geq 3$ and the configuration is in general position (in particular, the points are not all collinear). Suppose $g \cdot \mathbf{x} = \mathbf{x}$ for some $g = (t, R) \in G$ and $\mathbf{x} \in X_{\text{gp}}$. 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:

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

We consider two cases for $R \in \mathrm{O}(2)$:

Case 1: $R \in \mathrm{SO}(2)$ (rotation). For $R \neq I$, the matrix $(R - I)$ has eigenvalues $e^{i\theta} - 1$ and $e^{-i\theta} - 1$ for rotation angle $\theta \neq 0$, giving rank 2 with trivial kernel. Thus $(R - I)v = 0$ for non-zero v implies $R = I$.

Case 2: $R \in \mathrm{O}(2) \setminus \mathrm{SO}(2)$ (reflection). A reflection has eigenvalues $+1$ and -1, so $(R - I)$ has rank 1 with a 1-dimensional kernel (the reflection axis). For $(R - I)v = 0$, the vector v must lie along this axis. However, for configurations in general position (which holds almost surely for randomly sampled cities), the set of difference vectors is not contained in any 1D subspace, i.e., the points are not all collinear. Thus, there exist pairs (i, j) and (k, l) such that the difference vectors are linearly independent, and no reflection can satisfy $(R - I)v = 0$ for both. Hence $R = I$.

With $R = I$, the condition becomes $t = 0$, so $g = e$ is the identity. Therefore, the action is free on configurations in general position with $n \geq 3$.

The action of E(2) on X is proper (it is an isometric Lie group action on $\mathbb{R}^{2n}$). By the quotient manifold theorem (Lee, 2003), 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:

$$\dim(X_{\mathrm{gp}}/G) = \dim(X) - \dim(G) = 2n - 3$$

Part (ii): Let $F : X_{\mathrm{gp}} \to Y$ be G-invariant (equivalently, G-equivariant with the trivial representation), meaning $F(g \cdot \mathbf{x}) = F(\mathbf{x})$ for all $g \in G$ and $\mathbf{x} \in X_{\mathrm{gp}}$.

For TSP edge prediction, we typically have $Y = \{0, 1\}^{n \times n}$ (adjacency matrices) and the representation is trivial, so the invariance condition holds.

Define $\widetilde{F} : X_{\mathrm{gp}}/G \to Y$ 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 invariance: 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 $[\mathbf{x}] \in X_{\mathrm{gp}}/G$, 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): The factorization $F = \widetilde{F} \circ \pi$ establishes a bijection between:

$$\begin{gathered}\{G\text{-invariant functions } X_{\mathrm{gp}} \to Y\} \longleftrightarrow \{\text{functions } X_{\mathrm{gp}}/G \to Y\} \\ F \longmapsto \widetilde{F} \\ \widetilde{F} \circ \pi \longleftarrow \widetilde{F}\end{gathered}$$

Therefore, learning any G-invariant function F is equivalent to learning the corresponding function $\widetilde{F}$ on the quotient manifold. Since X_{gp}/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_{gp}/G requires parametrization by $2n - 3$ variables
- Local charts for X_{gp}/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)-invariant learning reduces to learning on a lower-dimensional manifold, providing the theoretical foundation for improved sample efficiency. □

E.3. 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 $g \in E(2)$, 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 learned invariant embeddings) - 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):

$$\begin{aligned}
\Delta(g \cdot x_i) &= \alpha \sum_{j \neq i} w_{ij} \cdot \frac{g \cdot x_j - g \cdot x_i}{\|g \cdot x_j - g \cdot x_i\|_2} && (14)\\
&= \alpha \sum_{j \neq i} w_{ij} \cdot \frac{g(x_j - x_i)}{\|x_j - x_i\|_2} && (15)\\
&= g \cdot \left(\alpha \sum_{j \neq i} w_{ij} \cdot \frac{x_j - x_i}{\|x_j - x_i\|_2} \right) && (16)\\
&= g \cdot \Delta x_i && (17)
\end{aligned}$$

(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). □

F. Extended Related Work

F.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., 2016) 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., 2018), 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. Recent RL innovations include preference-based training methods: PO (Pan et al., 2025a) transforms rewards into pairwise preferences to accelerate learning on TSP/CVRP, while BOPO (Liao et al., 2025) employs best-anchored preference optimization to reduce optimality gaps on TSP. 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).

F.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) (Luo et al., 2023) 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 (Ye 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. Emerging directions include problem unification and multi-task learning: UniCO (Pan et al., 2025b) proposes reducing various COPs (ATSP, Hamiltonian cycle, SAT) to a matrix-encoded general TSP formulation, while GOAL (Drakulic et al., 2024) introduces the first generalist model that jointly solves multiple COPs (routing, scheduling, graph problems) using task-specific adapters, achieving competitive performance across diverse problem families. These methods demonstrate that architectural innovations can sometimes overcome the data requirements that limit standard approaches.

F.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 (Eijkelboom et al., 2024) and discrete flow matching (Gat et al., 2024), 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 (Qin et al., 2024) 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 (Feng & Yang, 2025) shows how continuous-time formulations avoid local optima more effectively than discrete-time approaches.

F.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., 2018) establishes fundamental representation limits, while work on algorithmic alignment (Xu et al., 2019) 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.

F.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. NeuroLKH (Xin et al., 2021b) integrates neural networks with LKH-3 to learn edge scores that guide the local search process, achieving strong performance on large-scale TSP. 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 (Pirnay & Grimm, 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).

G. Architecture Details

G.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.

G.2. Feature Representations and Initialization

The model maintains three distinct feature types throughout the network:

Node Features. Node embeddings $\mathbf{h} \in \mathbb{R}^{n\times 64}$ are initialized as learnable parameters, ensuring rotational invariance from the start. City coordinates $\mathbf{x} \in \mathbb{R}^{n\times 2}$ are maintained separately and used only to compute pairwise distances (which are rotationally invariant) in the message passing mechanism. Coordinates evolve through equivariant updates during message passing but never directly enter node feature computations.

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.

G.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:

$$\mathbf{h}_i \leftarrow \text{LN}(\mathbf{h}_i + f_{\text{node}}(\mathbf{h}_i, \sum_j \mathbf{m}_{ij})) \tag{18}$$

$$\mathbf{e}_{ij} \leftarrow \text{LN}(\mathbf{e}_{ij} + f_{\text{edge}}(\mathbf{e}_{ij}, \mathbf{m}_{ij}) + f_{\text{time}}(\mathbf{t})) \tag{19}$$

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

G.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.

H. Additional Experiment Details

H.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)

Sampling-Based Decoding For SL+S+2O results (Sampling with 2-opt), we use a sampling budget of 128 samples per instance, following the protocol from Sun & Yang (2023); Li et al. (2023). For each instance, 128 independent tours are generated using the diffusion model, 2-opt local search is applied to each tour, and the best tour (minimum length) is selected as the final solution. For TSPLIB experiments, we use 4× sampling (4 samples per instance) following Li et al. (2023) for fair comparison. The reported times include the full sampling and 2-opt processing.

H.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. For fair timing comparisons, direct diffusion-based baseline methods (DIFUSCO, T2T, Fast-T2T, CADO) were re-evaluated on our hardware using their official open-source implementations with default configurations. Other methods without available checkpoints (AM, POMO, GLOP, etc) use times reported in the original papers, which may have been measured on different hardware.

H.3. Randomness in Results

Due to the stochastic nature of diffusion models, all results reported are averaged over five runs with different random seeds. We observe low variance within $\pm 0.03\%$ across all runs.

H.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., 2018) 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 reduces the computational complexity from O(n²) to O(nk) while preserving the most relevant edges for tour construction.

EGNN Integration with Sparse Graphs The EGNN architecture naturally accommodates sparse graphs through its message-passing formulation. For sparse graphs, messages are only computed and aggregated along existing edges (Eq. 8), and coordinate updates (Eq. 9) sum only over connected neighbors. Importantly, E(2)-equivariance is preserved under sparsification because: (1) k-nearest neighbor selection is invariant to rotations, translations, and reflections—if city j is among city i's k nearest neighbors, this relationship is preserved under any E(2) transformation; (2) all EGNN operations (distance computation, message aggregation, coordinate updates) operate on the same neighbor sets regardless of coordinate frame. The diffusion process remains identical, with edge features $\mathbf{e}_{ij}$ initialized and updated only for edges in the sparse graph. Dense matrix operations are replaced with their sparse equivalents throughout the network.

H.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.

H.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 training objective minimizes weighted cross-entropy:

$$\mathcal{L}(\theta) = \mathbb{E}_{t\sim\mathcal{U}(0,1),\mathbf{X}_0\sim p_{\text{data}},\mathbf{X}_t\sim q(\mathbf{X}_t|\mathbf{X}_0)}\left[(1-\sqrt{t})\cdot \text{CE}(s_\theta(\mathbf{X}_t, t), \mathbf{X}_0)\right] \tag{20}$$

The weighting $(1 - \sqrt{t})$ 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.

I. Additional Results

I.1. Multi-Step Methods for Categorical CTMC

A key question is how numerical methods like PNDM and DEIS, originally developed for continuous Gaussian diffusion, can be applied to categorical continuous-time Markov chains (CTMCs). Unlike continuous diffusion where the probability flow ODE can be directly integrated, categorical states require sampling from discrete distributions. We address this by decomposing the reverse process into two components: (1) multi-step prediction smoothing and (2) exact CTMC posterior sampling.

Prediction Smoothing via Adams-Bashforth Multi-step methods improve $\mathbf{X}_0$ prediction by combining estimates from multiple timesteps using Adams-Bashforth coefficients. For a 4th-order method (PNDM), given predictions $\hat{\mathbf{X}}_0^{(t_1)}, \ldots, \hat{\mathbf{X}}_0^{(t_4)}$ at recent timesteps:

$$\tilde{p}(\mathbf{X}_0) = \frac{55\hat{p}^{(t_1)} - 59\hat{p}^{(t_2)} + 37\hat{p}^{(t_3)} - 9\hat{p}^{(t_4)}}{24} \tag{21}$$

This extrapolation reduces prediction variance by leveraging temporal coherence, analogous to momentum in optimization.

Exact CTMC Posterior Sampling Given the smoothed prediction $\tilde{p}(\mathbf{X}_0 = 1|\mathbf{X}_t)$, we sample the next state from the exact posterior using Bayes' rule. For binary CTMCs, the posterior is:

$$q(\mathbf{X}_s = 1|\mathbf{X}_t, \tilde{p}) = \frac{P(\mathbf{X}_t|\mathbf{X}_s = 1) \cdot \mathbb{E}_{\tilde{p}}[P(\mathbf{X}_s = 1|\mathbf{X}_0)]}{\sum_{k\in\{0,1\}} P(\mathbf{X}_t|\mathbf{X}_s = k) \cdot \mathbb{E}_{\tilde{p}}[P(\mathbf{X}_s = k|\mathbf{X}_0)]} \tag{22}$$

where $P(\mathbf{X}_t|\mathbf{X}_s)$ uses the closed-form transition probabilities from Eq. 3. The expectation $\mathbb{E}_{\tilde{p}}[\cdot]$ is taken over the predicted distribution. This ensures mathematically correct reverse dynamics while benefiting from improved predictions.

Integration Each reverse step proceeds as: (1) compute $\hat{p}(\mathbf{X}_0|\mathbf{X}_t)$ via the neural network, (2) apply Adams-Bashforth smoothing to obtain $\tilde{p}$, (3) sample $\mathbf{X}_s \sim q(\mathbf{X}_s|\mathbf{X}_t, \tilde{p})$ from the exact posterior. This approach preserves the CTMC structure while accelerating convergence through multi-step prediction.

I.2. 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 multi-step methods described in Section I.1 that can achieve better speed-quality trade-offs than discrete-time approaches. We evaluate 12 different solver configurations ranging from classical first-order methods to modern multi-step schemes.

Table 7 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.

Table 7. 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	$\sim$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., 2020a)	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, 2022)	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., 2022)	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., 2022)	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

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.

I.3. Full TSP Results

This section presents the complete comparison on all TSP benchmarks, including all baseline methods and decoding strategies. Table 8 shows results for TSP-50 and TSP-100, Table 9 shows results for TSP-500 and TSP-1000, and Table 10 shows results for TSP-10000.

Table 8. Full results on TSP-50 and TSP-100. RL: Reinforcement Learning, SL: Supervised Learning, G: Greedy Decoding, S: Sampling, BS: Beam Search, 2O: 2-opt Post-processing. Concorde* represents the baseline for computing the gap. Bold: best, underlined: second-best.

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., 2018)	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%
DISCO (Zhao et al., 2024)	SL+G	5.70	0.16%	7.80	0.58%
T2T (Li et al., 2023)	SL+G	5.69	0.04%	7.77	0.18%
Fast T2T (Ts=5) (Li et al., 2024)	SL+G	5.69	0.02%	7.76	0.07%
BQ-perceiver (Drakulic et al., 2023)	SL+G	–	–	7.84	0.97%
BQ-transformer (Drakulic et al., 2023)	SL+G	–	–	7.79	0.35%
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%**
EDISCO with 5-step DEIS-2 (ours)	SL+G	5.69	0.02%	7.76	0.06%
AM (Kool et al., 2018)	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%
DISCO (Zhao et al., 2024)	SL+S	5.69	0.00%	7.76	0.03%
T2T (Li et al., 2023)	SL+G+2O	5.69	0.02%	7.76	0.06%
Fast T2T (Ts=3,Tg=3) (Li et al., 2024)	SL+G+2O	5.69	0.01%	7.76	0.03%
BQ-transformer (bs16) (Drakulic et al., 2023)	SL+BS	–	–	7.76	0.01%
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%**
EDISCO with 5-step DEIS-2 (ours)	SL+G+2O	5.69	0.01%	7.76	0.02%

Table 8 presents the complete comparison on small-scale TSP instances. On TSP-50, EDISCO achieves near-optimal performance with 0.01% gap using greedy decoding, matching CADO while substantially outperforming DIFUSCO (0.48%) and other baselines. On TSP-100, EDISCO achieves 0.04% gap, improving over DIFUSCO (1.01%), T2T (0.18%), and CADO (0.08%). With 2-opt post-processing, EDISCO reaches optimal or near-optimal solutions (0.00%/0.01% gaps), matching the best results of CADO and BQ-transformer. Notably, EDISCO achieves these results using only supervised learning, without requiring the additional reinforcement learning fine-tuning used by CADO.

Table 9. Full 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. Bold: best, underlined: second-best.

Algorithm	Type	TSP-500			TSP-1000		
		Length↓	Gap↓	Time	Length↓	Gap↓	Time
Concorde* (Applegate et al., 2006)	Exact	16.55	–	37.66 m	23.12	–	6.65 h
Gurobi (Gurobi Optimization, LLC, 2020)	Exact	16.55	0.00%	45.63 h	–	–	–
LKH-3 (default) (Helsgaun, 2017)	Heuristics	16.55	0.00%	46.28 m	23.12	0.00%	2.57 h
AM (Kool et al., 2018)	RL+G	20.02	20.99%	1.51 m	31.15	34.75%	3.18 m
GCN (Joshi et al., 2019)	SL+G	29.72	79.61%	6.67 m	48.62	110.29%	28.52 m
POMO+EAS-Emb (Kwon et al., 2020)	RL+AS+G	19.24	16.25%	12.80 h	–	–	–
POMO+EAS-Tab (Kwon et al., 2020)	RL+AS+G	24.54	48.22%	11.61 h	49.56	114.36%	63.45 h
DIMES (Qiu et al., 2022)	RL+G	18.93	14.38%	0.97 m	26.58	14.97%	2.08 m
DIMES (Qiu et al., 2022)	RL+AS+G	17.81	7.61%	2.10 h	24.91	7.74%	4.49 h
DIFUSCO (Sun & Yang, 2023)	SL+G	18.11	9.41%	5.70 m	25.72	11.24%	17.33 m
T2T (Li et al., 2023)	SL+G	17.39	5.09%	4.90 m	25.17	8.87%	15.66 m
Fast T2T (Ts=5) (Li et al., 2024)	SL+G	17.53	5.94%	0.37 m	24.57	6.29%	1.35 m
BQ-perceiver (Drakulic et al., 2023)	SL+G	17.41	5.22%	**0.13 m**	25.19	8.97%	**0.37 m**
CADO (Yoon et al., 2024)	SL+RL+G	16.93	2.30%	8.23 m	23.89	3.33%	18.42 m
EDISCO with 50-step PNDM (ours)	SL+G	**16.87**	**1.95%**	2.19 m	**23.78**	**2.85%**	6.84 m
EDISCO with 5-step DEIS-2 (ours)	SL+G	17.01	2.78%	0.23 m	24.83	4.42%	0.75 m
DIMES (Qiu et al., 2022)	RL+G+2O	17.65	6.62%	1.01 m	24.83	7.38%	2.29 m
DIMES (Qiu et al., 2022)	RL+AS+G+2O	17.31	4.57%	2.10 h	24.33	5.22%	4.49 h
DIFUSCO (Sun & Yang, 2023)	SL+G+2O	16.81	1.55%	5.75 m	23.55	1.86%	17.52 m
T2T (Li et al., 2023)	SL+G+2O	16.68	0.78%	4.98 m	23.41	1.25%	15.90 m
Fast T2T (Ts=5,Tg=5) (Li et al., 2024)	SL+G+2O	16.61	0.39%	2.17 m	23.25	0.58%	8.62 m
BQ-transformer (Drakulic et al., 2023)	SL+G	16.75	1.18%	**0.25 m**	23.65	2.29%	**0.50 m**
CADO (Yoon et al., 2024)	SL+RL+G+2O	16.59	0.24%	8.35 m	23.28	0.69%	18.67 m
EDISCO with 50-step PNDM (ours)	SL+G+2O	**16.58**	**0.18%**	2.35 m	**23.24**	**0.52%**	6.97 m
EDISCO with 5-step DEIS-2 (ours)	SL+G+2O	16.59	0.26%	0.40 m	23.31	0.82%	0.90 m
EAN (Deudon et al., 2018)	RL+S+2O	23.75	43.57%	57.76 m	47.73	106.46%	5.39 h
AM (Kool et al., 2018)	RL+BS	19.53	18.03%	21.99 m	29.90	29.23%	1.64 h
GCN (Joshi et al., 2019)	SL+BS	30.37	83.55%	38.02 m	51.26	121.73%	51.67 m
DIMES (Qiu et al., 2022)	RL+S	18.84	13.84%	1.06 m	26.36	14.01%	**2.38 m**
DIMES (Qiu et al., 2022)	RL+AS+S	17.80	7.55%	2.11 h	24.89	7.70%	4.53 h
DIFUSCO (Sun & Yang, 2023)	SL+S	17.48	5.65%	19.02 m	25.11	8.61%	59.18 m
T2T (Li et al., 2023)	SL+S	17.02	2.84%	15.98 m	24.72	6.92%	53.92 m
Fast T2T (Ts=5) (Li et al., 2024)	SL+S	17.02	2.85%	1.12 m	24.07	4.10%	4.65 m
CADO (Yoon et al., 2024)	SL+RL+S	16.76	1.27%	26.89 m	23.67	2.38%	61.23 m
EDISCO with 50-step PNDM (ours)	SL+S	**16.72**	**1.05%**	7.82 m	**23.57**	**1.95%**	23.27 m
EDISCO with 5-step DEIS-2 (ours)	SL+S	16.80	1.50%	**0.85 m**	23.82	3.02%	2.58 m
DIMES (Qiu et al., 2022)	RL+S+2O	17.64	6.56%	1.10 m	24.81	7.29%	2.86 m
DIMES (Qiu et al., 2022)	RL+AS+S+2O	17.29	4.48%	2.11 h	24.32	5.17%	4.53 h
DIFUSCO (Sun & Yang, 2023)	SL+S+2O	16.69	0.83%	19.05 m	23.42	1.30%	59.53 m
T2T (Li et al., 2023)	SL+S+2O	16.61	0.37%	16.03 m	23.30	0.78%	54.67 m
Fast T2T (Ts=5,Tg=5) (Li et al., 2024)	SL+S+2O	16.58	0.21%	6.85 m	23.22	0.42%	18.28 m
BQ-transformer (bs16) (Drakulic et al., 2023)	SL+S+2O	16.57	0.15%	18.00 m	23.20	0.35%	42.00 m
CADO (Yoon et al., 2024)	SL+RL+S+2O	16.57	0.12%	27.01 m	23.19	0.30%	61.48 m
EDISCO with 50-step PNDM (ours)	SL+S+2O	**16.56**	**0.08%**	8.03 m	**23.17**	**0.22%**	23.48 m
EDISCO with 5-step DEIS-2 (ours)	SL+S+2O	16.57	0.12%	**1.00 m**	23.20	0.35%	**2.80 m**

Table 9 presents comprehensive results on large-scale TSP instances. With greedy decoding, EDISCO achieves 1.95% and 2.85% gaps on TSP-500 and TSP-1000 respectively, significantly outperforming DIFUSCO (9.41%, 11.24%), T2T (5.09%, 8.87%), and CADO (2.30%, 3.33%) while being 3.8× and 2.7× faster than CADO. The fast EDISCO-DEIS2 variant achieves competitive gaps (2.78%, 4.42%) in only 0.23 and 0.75 minutes, enabling real-time applications.

With 2-opt post-processing, EDISCO achieves 0.18% and 0.52% gaps, outperforming CADO (0.24%, 0.69%) while being 3.6× and 2.7× faster. Under sampling-based decoding, EDISCO achieves 1.05% and 1.95% gaps compared to DIFUSCO's 5.65% and 8.61%. With sampling plus 2-opt, EDISCO reaches state-of-the-art gaps of 0.08% and 0.22% in 8.03 and 23.48 minutes, compared to CADO's 0.12% and 0.30% in 27.01 and 61.48 minutes, representing both improved solution quality and 3.4×/2.6× speedups.

Table 10. 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	–	8.8 h
LKH-3 (less trials) (Helsgaun, 2017)	Heuristics	71.79	0.03%	51.27 m
2-opt (Lin & Kernighan, 1973)	Heuristics	91.16	27.02%	28.49 m
Farthest Insertion	Heuristics	80.59	12.36%	13.25 m
AM (Kool et al., 2018)	RL+G	141.51	97.17%	7.68 m
GLOP (Ye et al., 2024)	RL+G	75.29	4.90%	1.90 m
DIMES (Qiu et al., 2022)	RL+AS+G	80.45	12.09%	3.07 h
DIFUSCO (Sun & Yang, 2023)	SL+G	78.35	8.95%	28.51 m
T2T (Li et al., 2023)	SL+G	73.87	2.92%	1.52 h
DISCO (Zhao et al., 2024)	SL+G	73.85	2.90%	1.52 h
EDISCO with 50-step PNDM (ours)	SL+G	**73.19**	**1.98%**	12.18 m
EDISCO with 5-step DEIS-2 (ours)	SL+G	74.08	3.21%	**1.28 m**
DIFUSCO (Sun & Yang, 2023)	SL+G+2O	73.99	3.10%	35.38 m
EDISCO with 50-step PNDM (ours)	SL+G+2O	**72.87**	**1.53%**	12.72 m
EDISCO with 5-step DEIS-2 (ours)	SL+G+2O	73.54	2.47%	**1.48 m**
AM (Kool et al., 2018)	RL+BS	129.40	80.28%	1.81 h
GLOP (Ye et al., 2024)	RL+S	75.27	4.88%	5.96 m
DIFUSCO (Sun & Yang, 2023)	SL+S	95.52	33.09%	6.59 h
T2T (Li et al., 2023)	SL+S	73.81	2.84%	2.47 h
DISCO (Zhao et al., 2024)	SL+S	73.81	2.84%	48.77 m
EDISCO with 50-step PNDM (ours)	SL+S	**72.77**	**1.39%**	38.92 m
EDISCO with 5-step DEIS-2 (ours)	SL+S	73.37	2.23%	**4.15 m**
DIFUSCO (Sun & Yang, 2023)	SL+S+2O	74.66	4.03%	6.67 h
DISCO (Zhao et al., 2024)	SL+GS+MCTS	73.69	2.68%	2.1 h
EDISCO with 50-step PNDM (ours)	SL+S+2O	**72.63**	**1.20%**	39.28 m
EDISCO with 5-step DEIS-2 (ours)	SL+S+2O	73.15	1.92%	**4.58 m**

Table 10 presents results on the challenging TSP-10000 benchmark, demonstrating EDISCO's scalability to very large problem instances. With greedy decoding, EDISCO with 50-step PNDM achieves a 1.98% optimality 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. For ultra-fast inference, EDISCO with 5-step DEIS-2 achieves 3.21% gap in only 1.28 minutes, representing a 9.5× speedup over the PNDM variant while still outperforming DIFUSCO.

When enhanced with 2-opt post-processing, EDISCO-PNDM achieves 1.53% gap in 12.72 minutes, substantially improving over DIFUSCO (3.10%, 35.38m), while EDISCO-DEIS2 reaches 2.47% in just 1.48 minutes. Under sampling-based decoding, EDISCO-PNDM achieves a 1.39% gap compared to DIFUSCO's 33.09% and outperforms both T2T and DISCO (2.84%) while being 3.8× faster than T2T; EDISCO-DEIS2 achieves 2.23% in 4.15 minutes. With sampling plus 2-opt, EDISCO-PNDM reaches a state-of-the-art 1.20% gap in 39.28 minutes, compared to DIFUSCO's 4.03% in 6.67 hours and DISCO's 2.68% in 2.1 hours, representing both a 3.4× improvement in solution quality over DIFUSCO and 10× speedup. EDISCO-DEIS2 achieves 1.92% gap in only 4.58 minutes, providing an excellent trade-off between solution quality and inference speed for time-sensitive applications.

I.4. 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 11 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).

Table 11. 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 Fast T2T are from Li et al. (2024). 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	Fast T2T	EDISCO (Ours)
eil51	16.767%	40.025%	1.725%	1.529%	0.314%	0.314%	**0.00%**	0.217%
berlin52	4.169%	33.225%	0.449%	0.142%	**0.000%**	**0.000%**	**0.00%**	**0.000%**
st70	1.737%	24.785%	0.040%	0.764%	0.172%	**0.000%**	**0.00%**	**0.000%**
eil76	1.992%	27.411%	0.096%	0.163%	0.217%	0.163%	**0.00%**	0.108%
pr76	0.816%	27.702%	1.228%	0.039%	0.043%	0.039%	**0.00%**	0.024%
rat99	2.645%	17.633%	0.123%	0.550%	0.016%	**0.000%**	**0.00%**	**0.000%**
kroA100	4.017%	28.828%	18.313%	0.728%	0.050%	**0.000%**	**0.00%**	**0.000%**
kroB100	5.142%	34.668%	1.119%	0.147%	0.006%	**0.000%**	0.65%	0.003%
kroC100	0.972%	35.506%	0.349%	1.571%	**0.000%**	**0.000%**	**0.00%**	**0.000%**
kroD100	2.717%	38.018%	0.866%	0.572%	**0.000%**	**0.000%**	**0.00%**	0.002%
kroE100	1.470%	26.568%	1.832%	0.140%	**0.000%**	**0.000%**	**0.00%**	**0.000%**
rd100	3.407%	50.432%	1.725%	0.003%	**0.000%**	**0.000%**	**0.00%**	0.001%
eil101	2.994%	26.701%	0.387%	1.529%	0.124%	**0.000%**	**0.00%**	0.008%
lin105	1.739%	34.902%	1.867%	0.484%	0.441%	0.393%	**0.00%**	0.267%
pr107	3.933%	80.564%	0.898%	0.439%	0.714%	0.155%	0.62%	**0.093%**
pr124	3.677%	70.146%	10.322%	0.755%	0.997%	0.584%	**0.08%**	0.372%
bier127	5.908%	45.561%	3.044%	1.948%	1.064%	0.718%	1.50%	**0.481%**
ch130	3.182%	39.090%	0.709%	3.519%	0.077%	0.077%	**0.00%**	0.046%
pr136	5.064%	58.673%	**0.000%**	3.387%	0.182%	**0.000%**	0.01%	0.004%
pr144	7.641%	55.837%	1.526%	3.581%	1.816%	**0.000%**	0.39%	0.011%
ch150	4.584%	49.743%	0.312%	2.113%	0.473%	0.324%	**0.00%**	0.218%
kroA150	3.784%	45.411%	0.724%	2.984%	0.193%	0.193%	**0.00%**	0.117%
kroB150	2.437%	56.743%	0.086%	3.258%	0.366%	0.021%	0.07%	**0.013%**
pr152	7.494%	33.925%	**0.029%**	3.119%	0.687%	0.687%	0.19%	0.428%
u159	7.551%	63.338%	10.534%	1.020%	**0.000%**	**0.000%**	**0.00%**	0.003%
rat195	6.893%	24.968%	0.743%	1.666%	0.887%	0.018%	0.79%	**0.012%**
d198	373.020%	62.351%	0.522%	4.772%	**0.000%**	**0.000%**	0.86%	0.006%
kroA200	7.106%	40.885%	1.441%	2.029%	0.259%	**0.000%**	0.49%	0.007%
kroB200	8.541%	43.643%	2.064%	2.589%	0.171%	0.171%	2.50%	**0.114%**
Mean	16.767%	40.025%	1.725%	1.529%	0.319%	0.133%	0.28%	**0.088%**

EDISCO achieves the lowest average optimality gap of 0.088%, representing a 33.8% 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%.

I.5. 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:

$$\beta(t) = \beta_{\min} + t(\beta_{\max} - \beta_{\min}) \tag{23}$$

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

$$\beta(t) = ab^t \log(b) \tag{24}$$

Cosine Schedule: Following Sun et al. (Sun et al., 2022), we tested cosine schedules with improved numerical stability:

$$\beta(t) = \text{clip}\left(\frac{\pi}{4} \cdot \frac{\tan(\pi t/2)}{\sqrt{\cos(\pi t/2) + \epsilon}}, \beta_{\min}, \beta_{\max}\right) \tag{25}$$

Table 12. 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	Configuration	Optimality Gap (%)		Conv.	Inference
Type		Best	Final	Epoch	Time (s)
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

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 12 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 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.

I.6. 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 $t = 0$.

Theoretical Foundation The proof of Proposition 3.1 is given in Appendix E.1. The key result is that the posterior variance Var$[X_0|X_t]$ scales linearly with t for small t, which motivates the linear mixing weight $w(t) = t$: at high t (unreliable predictions), we rely on CTMC dynamics; at low t (reliable predictions), we rely on direct prediction. The deterministic switching at $t < 0.1$ is justified by vanishing variance as $t \to 0$.

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

The linear weight function $w(t) = t$ achieves the best performance, providing a smooth transition from exploration

Table 13. 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

(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 14. 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†
$t < 0.05$	$\|\Delta t\| < 0.01$	0.04	0.09	2.41	3.2%
$t < 0.1$	$\|\Delta t\| < 0.02$	**0.01**	**0.04**	**1.95**	**0.0%**
$t < 0.15$	$\|\Delta t\| < 0.03$	0.02	0.05	1.98	0.0%
$t < 0.2$	$\|\Delta t\| < 0.04$	0.03	0.07	2.12	0.0%
$t < 0.25$	$\|\Delta t\| < 0.05$	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 15. 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.

I.7. 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 16. Effect of step size α on model performance and training stability (TSP-50). *Standard deviation of coordinates 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 16, 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 coordinates 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 17. 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 17 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 18. 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.

Discussion on Hyperparameter Robustness While our ablation studies reveal sensitivity to α and τ during initial design exploration—particularly the coordinate collapse phenomenon when $\alpha \geq 0.2$—we emphasize three important aspects of these architectural stabilizers: (1) **Optimal settings are robust**: Once identified, $\alpha = 0.1$ and $\tau = 10$ work consistently across all problem sizes and types (TSP-50/100/500/1000, CVRP-20/50/100, ESTP-10/20/50) without per-problem retuning, as demonstrated throughout our experiments. (2) **Comparable to standard practice**: This sensitivity is analogous to other architectural hyperparameters in deep learning (e.g., learning rates, attention heads, batch normalization momentum), which require careful tuning during model development but remain stable once configured. (3) **Strong generalization validates stability**: Our cross-distribution evaluation (Table 2 in the main text) demonstrates that these settings generalize well to out-of-distribution instances (Cluster, Explosion, Implosion), confirming they are not narrowly tuned to specific data distributions.

The thorough sensitivity analysis presented here provides practitioners with clear guidance for implementation and demonstrates our commitment to transparency in reporting both the strengths and limitations of our architectural design.

I.8. 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 19 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 19. 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	Training Instances (K) / Optimality Gap (%)				
	Parameters	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 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 is comparable to 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.

I.9. Training Resource Requirements

Table 20 presents comprehensive training resource requirements for EDISCO-Full, demonstrating its efficiency on commodity hardware. Unlike prior methods requiring multi-GPU setups (e.g., T2T uses 4× A100 GPUs with 1.5M training instances), EDISCO trains effectively on a single NVIDIA A6000 GPU through efficient graph sparsification and superior sample efficiency. The combination of E(2)-equivariance and continuous-time diffusion enables 3× better sample efficiency (500K vs 1.5M training instances for TSP-50/100), translating to proportionally reduced training time and memory footprint.

Graph sparsification (k-nearest neighbors with k=50 for TSP-500, k=100 for TSP-1000/10000) reduces memory complexity from $O(n^2)$ to $O(kn)$, enabling practical training on a single GPU even for TSP-10000. TSP-500/1000/10000 utilize curriculum learning initialized from smaller problem checkpoints, further improving sample efficiency. The peak memory usage remains well within the A6000's 48GB capacity across all problem scales, demonstrating EDISCO's practical accessibility compared to methods requiring expensive multi-GPU infrastructure.

Table 20. Training resource requirements for EDISCO-Full on a single NVIDIA A6000 GPU (48GB). Peak memory usage includes model parameters, optimizer states, and batch processing. EDISCO achieves state-of-the-art performance using 3× less training data than baseline methods while training on a single GPU.

Problem Scale	Dataset Size	Batch Size	Training Epochs	Training Time	Peak Memory
TSP-50	500K	64	100	18h 45m	19.2 GB
TSP-100	500K	32	100	68h 30m	26.8 GB
TSP-500	60K	16	50	31h 15m	23.5 GB
TSP-1000	30K	8	50	61h 20m	39.2 GB
TSP-10000	3K	4	50	17h 40m	44.8 GB

I.10. Visualization of EDISCO's E(2)-Equivariance

Figure 5 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 6 and 7. 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.

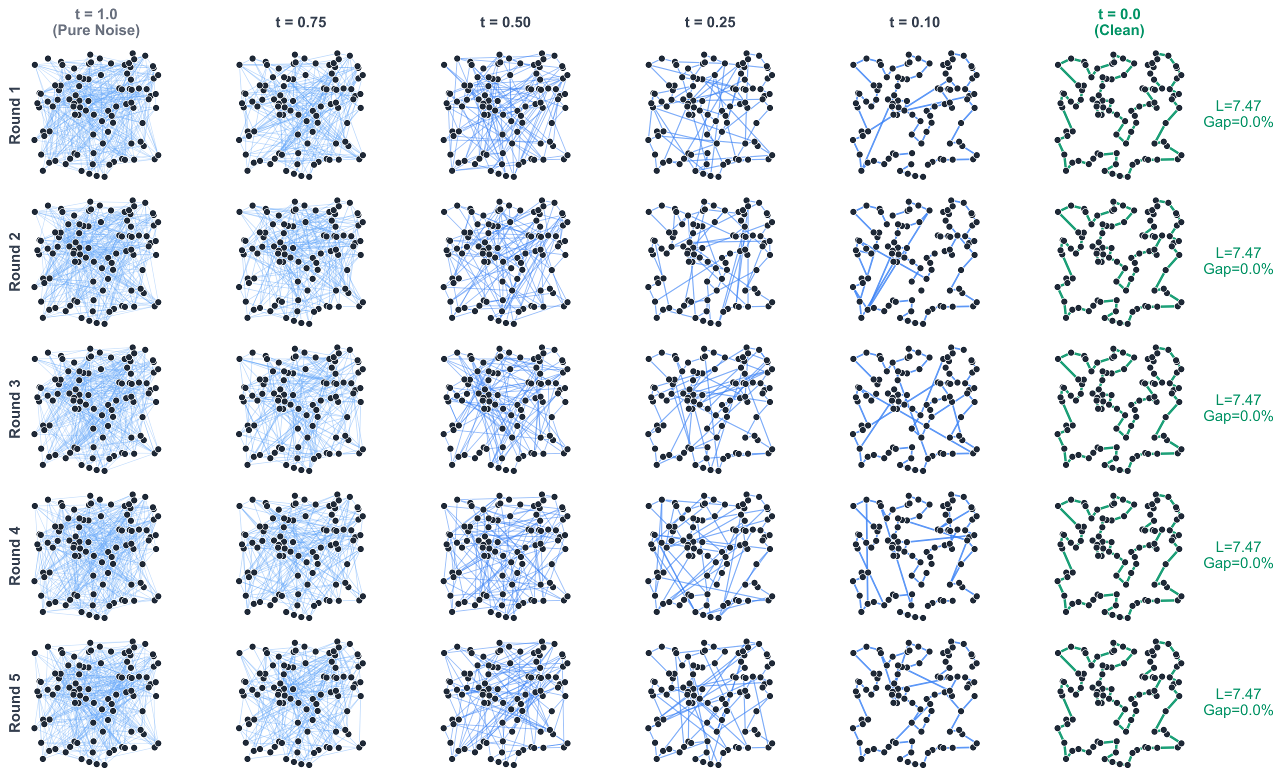

Figure 5. Five rounds of the denoising process of EDISCO on the original TSP instance.

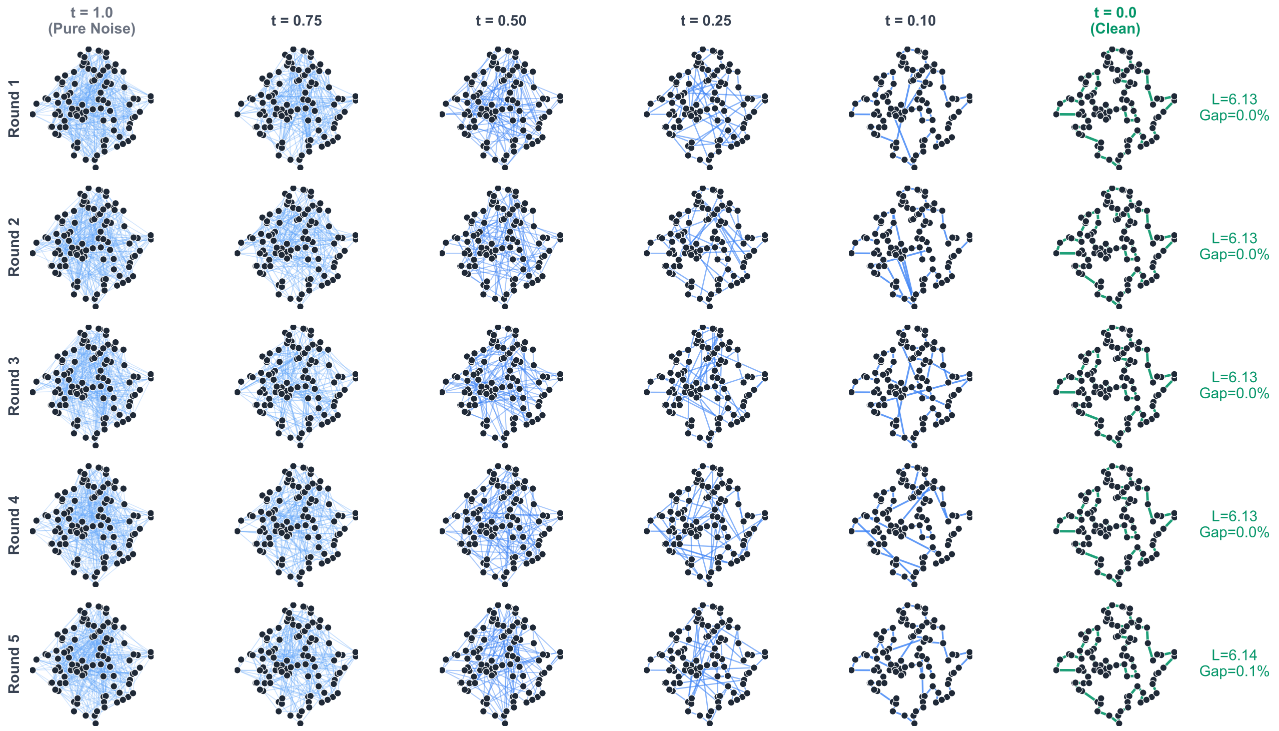

Figure 6. Five rounds of the denoising process of EDISCO on the 45° rotated instance.

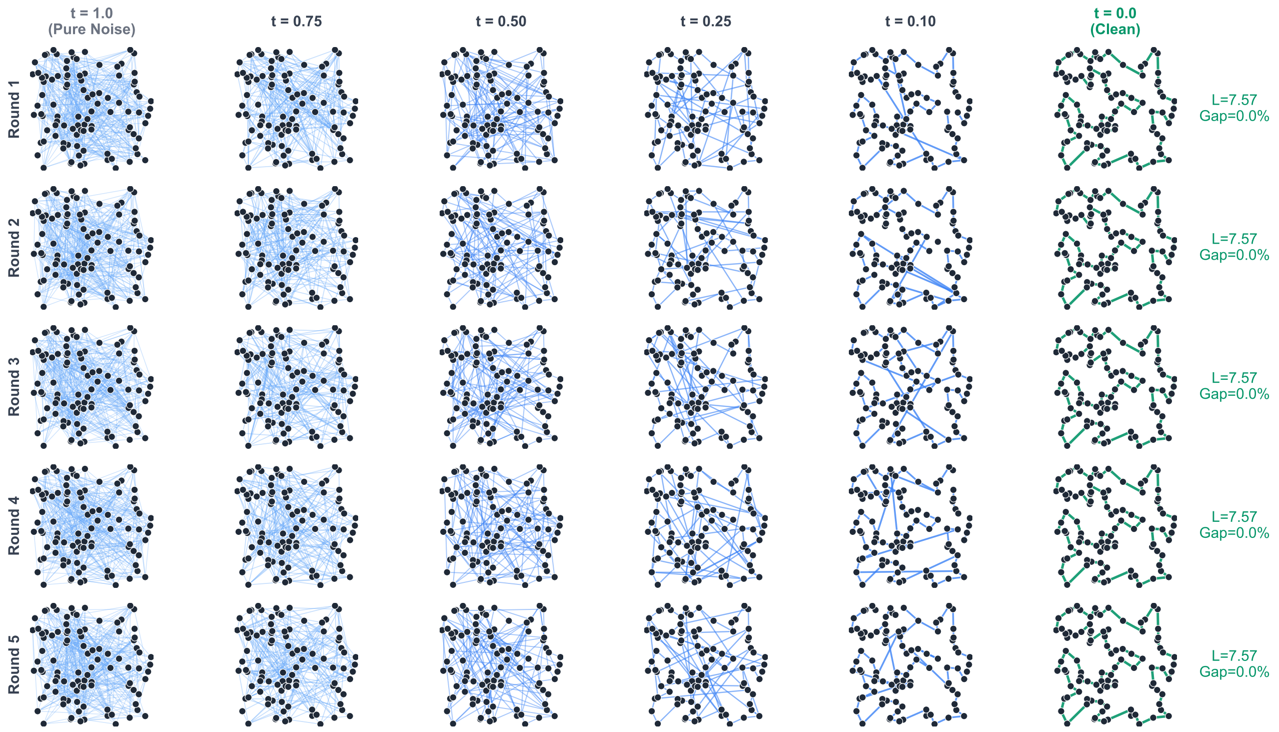

Figure 7. Five rounds of the denoising process of EDISCO on the 90° rotated instance.