Efficient Few-Step Solution Generation via Discrete Flow Matching for Combinatorial Optimization

Yuanshu Li[1], Di Wang[2], Wei Du[1], Xuan Wu[1], Peng Zhao[1], Yubin Xiao[1*], You Zhou [1*]

[1]Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education,
College of Computer Science and Technology,
Jilin University, China

[2]Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly,
Nanyang Technological University, Singapore

Abstract

Combinatorial optimization problems (COPs) are fundamental to many real-world applications where efficiently producing high-quality solutions is critical. Recent advances in diffusion-based non-autoregressive models have reformulated solving COPs as a generative process, achieving promising results. However, almost all of these methods still suffer from accumulated errors and high inference costs due to the multi-step stochastic denoising process. To address these issues, we propose EFLOCO, an efficient discrete flow matching method for solving COPs, learning structured and deterministic solution trajectories. EFLOCO replaces noise-driven updates with smooth and guided transitions, thereby improves inference stability and quality. Furthermore, we introduce an adaptive time-step scheduler that makes more efforts in critical transition regions, yielding strong performance under few-step constraints. Experiments on standard Traveling Salesman Problems (TSPs) and Asymmetric TSPs (ATSPs) show that our method consistently outperforms both learning-based and heuristic baselines in terms of solution quality and inference speed.

Code — https://github.com/liys-9/EFLOCO

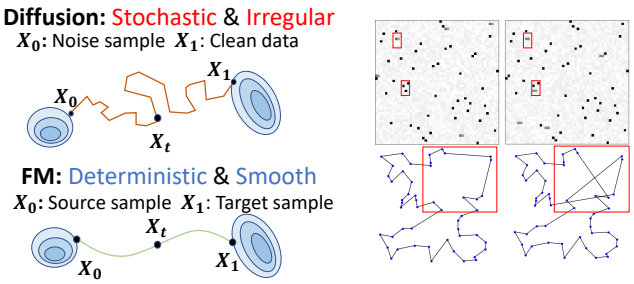
Introduction

Combinatorial Optimization Problems (COPs) aim to identify optimal solutions within discrete solution spaces, with numerous real-world applications in domains such as logistics (Kim et al. 2015). Given its significance, many exact and heuristic algorithms have been developed over the years (Applegate et al. 2007; Helsgaun 2017; Wu et al. 2025). Recently, deep learning-based models have emerged as a promising alternative for solving COPs (aka Neural Combinatorial Optimization, NCO), with significantly improved efficiency, generalization, and scalability (Bengio, Lodi, and Prouvost 2021; Wu et al. 2024; Xiao et al. 2025). Existing construction-based NCO models can be mainly categorized into autoregressive (AR) and non-autoregressive (NAR) frameworks. AR models generate solutions sequentially, where each decision depends on previous outputs (Kwon et al. 2020; Drakulic et al. 2023; Zhao et al. 2025a,b). In contrast, NAR models attempt to construct solutions in one shot for significantly improved inference speed (Joshi, Laurent, and Bresson 2019; Xiao et al. 2024; Wang et al. 2025).

Although early NAR models offered accelerated inference, they often yielded unsatisfactory solution quality (Joshi et al. 2021). Recent efforts have introduced discrete diffusion models as a promising paradigm to solve COPs. These models simulate a Markovian denoising process over discrete solution structures, gradually refining noisy intermediate solutions toward high-quality outputs (Graikos et al. 2022; Sun and Yang 2023). While diffusion-based methods outperform earlier NAR models, they present several key challenges in solving COPs. Typically, these methods construct solution trajectories by injecting random noise and progressively denoising intermediate states. However, the stochastic nature of this process often results in irregular and non-smooth transitions along the generative trajectory (Liu, Gong, and Liu 2022). These fluctuations tend to accumulate over time, destabilizing the trajectory and ultimately degrading the quality of the final solution (Lipman et al. 2023). Moreover, these models require long sampling chains to gradually approximate the target distribution, which leads to substantial computational overhead (Yang et al. 2023). This is particularly challenging in practical scenarios where rapid solution generation is required.

To address these issues, we propose an Efficient Discrete Flow Matching method for solving COPs (EFLOCO). We employ the flow matching (FM) paradigm to explicitly model discrete solution trajectories, avoiding the reliance on stochastic denoising. As illustrated in Figure 1(a), while diffusion models rely on long, stochastic sampling chains that are often unstable and computationally expensive, FM improves efficiency by learning direct, deterministic paths. Notably, despite FM's demonstrated improvement in generation speed and quality over diffusion models in computer vision tasks (Yang et al. 2024), directly applying FM to COPs presents unique challenges. Specifically, due to the discrete and sensitive nature of COPs, even small variations in intermediate representations (e.g., heatmaps for routing problems) can result in substantial deviations in final solutions, making trajectory coordination across multiple time steps

*Corresponding authors: xiaoyb21@mails.jlu.edu.cn, zyou@jlu.edu.cn.



(a) Diffusion vs. Flow matching (FM)

(b) Heatmaps and decoded solutions in FM

Figure 1: Sub-figure (a) compares diffusion models (top) and FM (bottom) in modeling distribution transformation paths. FM shows a more deterministic, smoother sampling path, reducing inference time and improving solution quality. In Sub-figure (b), the upper displays heatmaps of two solutions, while the lower visualizes the corresponding paths. As shown, although the heatmaps appear almost identical, the decoded solutions are significantly different.

particularly difficult (see Figure 1(b)). To mitigate this issue, we propose a velocity-based consistency loss that enforces smooth transitions across time steps, thereby improving trajectory stability and coherence, especially in few-step inference, where each step has a substantial impact.

Furthermore, we propose an adaptive time-step scheduling strategy (ATSS) that focuses on critical transition intervals during both training and inference, based on the structural variations along the path. This strategy enables the generation of higher-quality solutions within a limited number of steps. To the best of our knowledge, **EFLOCO is the first application of FM to solve COPs**, offering a novel learning-based method for efficiently generating high-quality solutions.

The key contributions of this work are as follows:

- We propose an efficient discrete flow matching method named EFLOCO that learns structured transition paths from a source distribution to target solutions, enabling fast and high-quality solution generation for COPs.
- We propose a velocity-based consistency loss to enhance trajectory stability and consistency on limited-step conditions, ensuring smooth velocity transitions across steps.
- We propose an adaptive time-step scheduling strategy (ATSS) based on local path variation, enabling the model to focus on critical transitions, which improves both sampling accuracy and efficiency in few-step inference.
- We demonstrate the excellent performance of EFLOCO, especially in inference speed, by conducting extensive experiments on Traveling Salesman Problems (TSPs) and Asymmetric TSPs (ATSPs).

Related Work

This section first reviews deep learning-based NCO methods, followed by a brief introduction to the FM framework.

Neural Network-based NCO Methods

Deep learning-based NCO methods have emerged as an impressive alternative for solving COPs, where the construction-based NCO methods can be broadly categorized into AR and NAR models based on their solution derivation process (Joshi et al. 2021). AR models (Drakulic et al. 2023), such as AM (Kool, van Hoof, and Welling 2019), POMO (Kwon et al. 2020), and LEHD (Luo et al. 2023), formulate solution derivation as a sequential decision process, where each step depends on the preceding outputs. While effective, their inherent decoding dependencies limit inference efficiency and search scalability (Drakulic et al. 2023; Luo et al. 2024; Gao et al. 2024). In contrast, NAR approaches leverage parallel or one-shot generation schemes to improve inference speed (Joshi, Laurent, and Bresson 2019; Xia et al. 2024). However, they often fail to match the solution quality achieved by AR counterparts (Xiao et al. 2024).

Recently, diffusion models have been introduced as a type of NAR models, leveraging their capacity to model complex solution distributions and improve generation quality (Graikos et al. 2022; Sun and Yang 2023). Nonetheless, their reliance on lengthy stochastic denoising chains restricts their ability to fully benefit from the efficiency advantages typically associated with NAR methods. Moreover, they exhibit limited controllability and stability, particularly under strict constraints on inference steps or runtime (Li et al. 2024).

Flow Matching (FM)

FM has emerged as a promising generative modeling framework that learns a velocity field to guide samples along a deterministic transformation from a source distribution to a target one. In contrast to diffusion models, which rely on multi-step stochastic denoising, FM enhances both efficiency and interpretability by directly modeling the transformation path (Lipman et al. 2023; Shaul et al. 2025). As shown in Figure 1(a), diffusion models follow stochastic, irregular, and often inefficient sampling trajectories, while FM produces smooth and deterministic paths that consistently map source samples to the target distribution. Although prior diffusion work (Ho, Jain, and Abbeel 2020) often denotes x_0 as the clean data without noise, we adopt the convention of most FM studies, i.e., x_0 as the noisy source-side distribution and x_1 as the clean target distribution, to avoid confusion and preserve consistency. Initially designed for continuous spaces, recent research has extended FM to discrete domains through Discrete FM, which explicitly models transition paths over finite state spaces. Discrete FM has demonstrated promising results in discrete-generation tasks such as code generation (Gat et al. 2024).

While FM has shown promising results in other domains, its direct application to solve COPs presents unique challenges. In particular, the discrete and globally coupled nature of solution spaces makes them highly sensitive to small changes that even minor perturbations in heatmaps can cause large deviations in final solutions. As shown in Figure 1(b), two slightly different heatmaps can lead to remarkably different solutions, highlighting the importance of ensuring stable trajectory coordination under few-step inference settings.

Methodology

In this section, we present EFLOCO, an efficient method for solving COPs via discrete flow matching.

Problem Definition

We consider a general class of COPs defined over discrete structures. Given an input $\mathcal{I}$, the goal is to find an optimal solution z^* that minimizes a task-specific cost function $\mathcal{C}$:

$$z^* = \arg\min_{z \in \mathcal{X}} \mathcal{C}(z; \mathcal{I}), \tag{1}$$

where $\mathcal{X}$ denotes the discrete solution space, such as permutations of nodes forming Hamiltonian cycles in TSP, or directed tours in ATSP, where edge costs are asymmetric.

Efficient Discrete Flow Matching for CO

Existing diffusion-based NCO approaches often rely on long sampling chains of local updates, which may destabilize during limited-step inference, leading to abrupt transitions that disrupt the structured evolution of solution trajectories. To address this, we develop an efficient discrete flow matching method named EFLOCO for solving COPs. We model the state transitions in COPs via a learned velocity field. By aligning each step with a globally consistent trajectory, EFLOCO avoids stochastic noise-driven updates and enables smooth, structure-aware solution evolution. This leads to more robust and efficient few-step inference, especially under constrained computation budgets. Such properties are especially important for solving COPs, where minor local perturbations can cause significant shifts in global solution quality (see Figure 1(b)).

EFLOCO generates discrete transition trajectories by interpolating between source and target solutions. We achieve this through four key components. Firstly, we construct a probability path to model intermediate distributions. Secondly, we learn a velocity field to guide state transitions along these paths. Then, we apply a discrete flow matching loss to supervise this process. Finally, we introduce a velocity-based consistency loss to ensure temporal coherence. An overview of this transition process is illustrated in Figure 2. We detail these components as follows:

Probability Path Construction To generate valid and high-quality solutions in discrete combinatorial spaces, we first define a family of probability paths $p_t(x)$, which interpolate between an initial distribution $p(x)$ and a target distribution $q(x)$ to model smooth transitions in discrete solution spaces. Each path is constructed by conditioning on a feasible target solution $x_1 \in \mathcal{S}$, where $\mathcal{S}$ denotes the set of valid structured configurations. For example, in the context of the Traveling Salesman Problems (TSPs), x_1 corresponds to a valid Hamiltonian tour represented as a binary adjacency matrix, while x_0 is initialized as a random binary matrix over all possible city connections, representing the unstructured starting state. The probability path $p_t(x)$ thus defines a continuous interpolation between these two distributions, progressively transforming a random configuration into a valid structured tour. These paths serve as the foundation for learning controllable, structure-aware solution trajectories, formulated as follows:

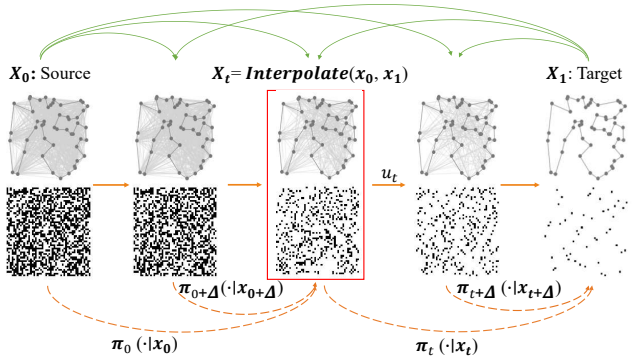


Figure 2: Illustration of EFLOCO's discrete trajectory generation. The source state x_0 and target state x_1 define the endpoints of a structured solution path. Intermediate states x_t are generated by interpolating along this path. At each time step, a velocity field u_t constructs a forward transition distribution π_t (see Eq. 8). To ensure temporal consistency, EFLOCO enforces alignment between adjacent distributions π_t and $\pi_{t+\Delta}$, promoting smooth and stable updates along the trajectory (see Eq. 10).

$$p_t(x) \triangleq \sum_{x_1 \in \mathcal{S}} p_t(x \mid x_1)\, q(x_1), \tag{2}$$

where solution $x \in \mathcal{S}$ corresponds to a structured representation of the target solution from $\mathcal{X}$ (e.g., a binary adjacency matrix as a heatmap representation for TSP).

We assume a factorized structure over dimensions, allowing the conditional probability distribution $p_t(x \mid x_1)$ to be decomposed as follows:

$$p_t(x \mid x_1) \triangleq \prod_{i=1}^{D} p_t(x_i \mid x_1^i), \tag{3}$$

where each component $p_t(x_i \mid x_i^1)$ interpolates between a simple base distribution $p_0(x_i^0)$ at time $t = 0$ and a delta distribution centered at the target value x_i^1 at time $t = 1$ and D denotes the dimensionality of the discrete representation of solution. This leads to the following boundary conditions:

$$p_0(x_i \mid x_i^1) = p(x_i), \quad p_1(x_i \mid x_i^1) = \delta_{x_i^1}(x_i), \tag{4}$$

where $\delta_{x_i^t}(x_i)$ denotes the Dirac delta function (Shaul et al. 2025) centered at the current state x_i^t, acting as a one-hot probability in the discrete space.

Following the discrete flow matching taxonomy (Shaul et al. 2025), we adopt the uniform-state formulation for path construction, which interpolates all dimensions simultaneously between the source and target states and assigns the time-dependent coefficients κ_j^t through the scheduler.

To increase the flexibility of path construction, we define the conditional probability path as a weighted combination

of basis transition components $w_j(x_i \mid x_0, x_1)$, with κ_j^t serving as the mixture weights:

$$p_t(x_i \mid x_0, x_1) = \sum_j \kappa_j^t \cdot w_j(x_i \mid x_0, x_1). \tag{5}$$

By modeling transition paths within a unified interpolation framework, EFLOCO learns to generate smooth, controllable, and structure-consistent trajectories in discrete spaces, thereby facilitating effective transitions toward solution spaces in COPs.

Velocity Field Modeling To capture the directional dynamics required for structured solution evolution in COPs, we introduce a probabilistic velocity field u_t that governs the evolution of each component x_i along the path. The transition from time step t to step $t + h$ is defined as follows:

$$x_i^{t+h} \sim \delta_{x_i^t} + h \cdot u_t^i(\cdot, x_t), \tag{6}$$

where the velocity field $u_t^i(\cdot, x_t)$ specifies a discrete transition distribution over next state candidates of x_i, conditioned on the complete current state x_t. In practice, the velocity field is approximated by the model's prediction of the final distribution at time $t = 1$. Specifically, we compute the transition tendency as follows:

$$u_t^i(x_i, x_t) = \frac{\dot{\kappa}_t}{1 - \kappa_t} \cdot \left[p_{1|t}(x_i \mid x_t) - \delta_{x_i^t}(x_i) \right], \tag{7}$$

where $p_{1|t}(x_i \mid x_t)$ denotes the model's predicted marginal distribution of x_i at the final step given the current state x_t, $\kappa_t \in [0, 1]$ denotes the time-dependent interpolation scheduler, and $\dot{\kappa}_t$ denotes the derivative of κ_t with respect to time, capturing the rate of change along the interpolation path.

This operation captures the directional deviation from the current state toward the anticipated target, serving as a learned guidance field for discrete trajectory construction.

Discrete Flow Matching Loss We train the model to align the predicted velocity field with the ground-truth velocity derived from the path p_t. We accomplish this alignment using forward Kullback-Leibler divergence as follows:

$$\mathcal{L}_{\mathrm{DFM}} := \mathbb{E}_{t \sim \rho(t)} \, \mathbb{E}_{z \sim p_t} \left[\mathrm{KL} \left(u_t(x) \parallel \hat{u}_t(x \mid z, t) \right) \right], \tag{8}$$

where $\rho(t)$ denotes the time-sampling distribution derived from the interpolation scheduler and $\hat{u}_t$ denotes the model's predicted velocity.

This objective encourages the model to learn discrete transition dynamics that adhere to the global path geometry between p and q, thereby facilitating efficient few-step sampling in discrete combinatorial spaces.

Velocity-based Consistency Loss In discrete COPs such as TSP, even minor inconsistencies in intermediate solutions can result in significant shifts in final outcomes due to the inherent sensitivity of discrete structures. To mitigate this instability, particularly in the context of few-step inference, we introduce a velocity-based consistency loss to promote smoother temporal dynamics by regularizing the behavior of predicted transitions across adjacent time steps.

Rather than directly enforcing similarity between velocity fields at adjacent steps, this objective focuses on the transition distributions they induce. Specifically, we assess whether the velocity fields learned at adjacent states lead to consistent transitions, thereby encouraging smoother and more coherent trajectory evolution.

Let $x_t \sim p_t$ and $x_{t+\Delta} \sim p_{t+\Delta}$ be two sampled intermediate states at nearby time steps. We define the model-induced transition distribution from state x_t as follows:

$$\pi_t(x' \mid x_t) := \hat{u}_t(x_t), \tag{9}$$

where $\pi_t(\cdot \mid x_t) \in \Delta^{|\mathcal{S}|}$ denotes a discrete probability distribution over possible next-step states. This formulation conceptualizes the predicted velocity field as a smooth, learnable probability flow over future configurations.

To promote coherent transitions, we define the symmetric consistency loss as follows:

$$\mathcal{L}_{\mathrm{cons}} = \mathbb{E}_{x_t, \Delta} \left[\| \pi_t(\cdot \mid x_t) - \pi_{t+\Delta}(\cdot \mid x_{t+\Delta}) \|_2^2 \right]. \tag{10}$$

This objective penalizes deviation between the induced next-step distributions from adjacent states and encourages structural smoothness in the learned dynamics, thereby improving the model's inference robustness within limited steps.

The total training objective combines the standard flow matching loss and the proposed consistency term:

$$\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{DFM}} + \lambda_{\mathrm{cons}} \cdot \mathcal{L}_{\mathrm{cons}}, \tag{11}$$

where λ_{cons} denotes a hyperparameter controlling the strength of the consistency regularization.

By promoting globally consistent and stable velocity fields, the combined training objective improves the quality of discrete solution trajectories, thereby enabling reliable few-step inference. This is particularly important in combinatorial tasks, where each inference step critically influences the structural validity and overall solution quality.

Adaptive Time-step Scheduling Strategy

In existing few-step discrete diffusion and flow matching frameworks, the time-step schedule is typically predefined using linear or cosine spacing. While easy to implement, they often fail to adapt to task-specific transition dynamics: often missing critical segments of the trajectory and limiting inference efficiency under step constraints (Xue et al. 2024). While (Xue et al. 2024) focus on critical time-step allocation during inference for continuous image diffusion, they pay little attention to its importance during training or to discrete problem settings.

Prior studies show that in many COP tasks, the evolution from the source to target distribution exhibits non-uniform transition patterns, with transitions concentrated in specific regions of the path (Seo 2025). Considering this, we introduce an adaptive time-step scheduling strategy (ATSS) that adapts to structural changes observed along the transition path, eliminating the need for manual time interval selection.

Firstly, we construct a surrogate integration error objective over a discrete trajectory and solve for a set of adaptive time steps that minimize this objective. The resulting

non-uniform transition patterns schedule allocates more resolution to high-variation regions along the path, improving both trajectory fidelity and sampling efficiency under limited step budgets. Formally, let the discrete trajectory be represented as a sequence of state vectors $\{x_{t_i}\}_{i=0}^{T}$, where $t_0 = 0$, $t_T = 1$, and t_i denotes a (monotonically increasing) time index. Let αt_i and σ_{t_i} be the interpolation weights assigned to the source and target states, respectively, at time t_i, as defined by a known scheduler (e.g., polynomial) (Shaul et al. 2025). We define the path weights' log-ratio at time t_i as:

$$\lambda_{t_i} := \log\left(\frac{\alpha_{t_i}}{\sigma_{t_i}}\right). \tag{12}$$

Then, we develop a path-aware time-step objective by estimating the local transition magnitude between states as:

$$\Delta_i := \left\|x_{t_i} - x_{t_{i-1}}\right\|, \quad \forall i \in= \{1, 2, \ldots, T\}, \tag{13}$$

where T denotes the number of time intervals used to discretize the trajectory. We aim to allocate time steps more effectively along the transition path by prioritizing regions with large state changes. To this end, we formulate the following discrete objective as an approximation of the integration error in continuous solvers:

$$\min_{\{\lambda_{t_1},\ldots,\lambda_{t_{T-1}}\}} \quad \sum_{i=1}^{T} \tilde{\varepsilon}_{t_i} \cdot e^{\lambda_{t_i}} \cdot \|x_{t_i} - x_{t_{i-1}}\|, \tag{14}$$

where the weight $\tilde{\varepsilon} t_i := \sigma {t_i}^p / \alpha_{t_i}$ reflects the relative complexity of the discrete path segment at time t_i, which is automatically determined by the interpolation scheduler employed to construct the flow matching trajectory. The exponential term $e^{\lambda_{t_i}}$ encourages finer resolution at later steps, where the influence of the source state diminishes. The term $\|x_{t_i} - x_{t_{i-1}}\|$ captures the actual state change. Together, these terms guide time step allocation toward segments with larger variation and higher learning demands.

Next, minimizing the objective in Eq. 14 results in a set of optimized log-interpolation ratios $\{\lambda_{t_i}\}$, which are subsequently mapped to time indices according to a predefined schedule (e.g., polynomial). This process produces a non-uniform discretization of the transition path, allocating finer resolution to segments with greater state variation, i.e., regions that are typically more informative and more difficult to model.

Finally, to incorporate this adaptive resolution into training, we define a non-uniform sampling distribution over time t using a piecewise-constant density based on the optimized intervals as follows:

$$\rho(t) \propto \sum_{n=1}^{N} \mathbf{1}_{[t_{n-1}, t_n)}(t) \cdot \frac{1}{t_n - t_{n-1}}. \tag{15}$$

This strategy ensures that each optimized interval is sampled uniformly, while high-resolution regions receive more probability mass overall. As a result, the model receives denser supervision in complex segments of the trajectory, improving convergence and inference quality, especially under few-step constraints that are pervasive in discrete combinatorial tasks such as TSP and ATSP.

The algorithm of ATSS is presented in Algorithm 1.

Algorithm 1: Adaptive Time-Step Scheduling (ATSS)

Require: Target trajectory $\{x_t\}_{t\in[0,1]}$, step number T, time scheduler $\kappa(t)$ and its derivative, time indices $\{t_i\}_{i=0}^{T}$.

Set initial time grid: $t_0 = 0$, $t_T = 1$;
for i in $1, ..., T-1$ **do**
  $t_i \leftarrow i/T$ *# Uniform spacing as baseline*
end for
 # Estimate local transition magnitude along path
for i in $1, ..., T$ **do**
  State change: $\Delta_i \leftarrow \|x_{t_i} - x_{t_{i-1}}\|$ (see Eq. 13)
  Path weight: $\tilde{\varepsilon}_{t_i} \leftarrow \sigma_{t_i}^p / \alpha_{t_i}$ (see Eq. 12)
  Cost: $\mathcal{L}_i \leftarrow \tilde{\varepsilon}_{t_i} \cdot e^{\lambda_{t_i}} \cdot \Delta_i$ (see Eq. 14)
end for
 # Optimize log-interpolation ratios for time allocation
$\{\lambda_{t_i}\}_{i=1}^{T-1} \leftarrow$ minimizing $\sum_{i=1}^{T} \mathcal{L}_i$ *# Allocate more resolution to high-change segments*
 # Convert optimized interpolation ratios to time indices
for i in $1, ..., T-1$ **do**
  $t_i \leftarrow \kappa^{-1}(e^{-\lambda_{t_i}})$
end for
return $\{t_0, t_1, \ldots, t_T\}$ *# Final adaptive time steps*

Experiments

This section provides a comprehensive evaluation of the proposed EFLOCO method.

Experimental Settings

While EFLOCO is generically applicable to various COPs, our evaluations primarily focus on TSP and ATSP, as these two represent common challenges within the NCO community, with well-established competitors that provide a robust benchmark. Performance is assessed based on solution quality (measured by optimality gap) and inference efficiency (measured by wall-clock time).

Datasets For TSP, we generate and label the training instances using Concorde for TSP-20/50/100 and LKH-3 for TSP-500/1000, following (Sun and Yang 2023; Li et al. 2023). The test datasets are generated using the same settings and instance counts as those in previous studies, with 1280 TSP-50/100 following (Joshi et al. 2021) and 128 TSP-500/1000 following (Fu, Qiu, and Zha 2021).

For ATSP, we generate and label the training instances using CPLEX for ATSP-20/50/100, following (Kwon et al. 2021). For testing, we randomly generate 1280 ATSP instances using the standard generation process.

Model Settings We adopt an anisotropic graph neural network architecture to jointly encode both node and edge features, following DIFUSCO (Sun and Yang 2023). Detailed network architecture and parameter settings are provided in Appendix A uploaded to our online code repository.

Baselines For TSP, we conduct extensive comparisons of EFLOCO against various baselines, including the exact solver Concorde (Applegate et al. 2006), 2-opt heuristic (Lin and Kernighan 1973), and a range of state-of-the-art (SOTA) learning-based approaches. The learning-based

Method (venue)	Type	TSP-20		TSP-50		TSP-100	
		Gap(%) ↓	Time ↓	Gap(%) ↓	Time ↓	Gap(%) ↓	Time ↓
Concorde	Exact	0.00	-	0.00	-	0.00	-
2-opt	Heuristic	2.51	8.41s	5.89	24.54s	7.61	53.74s
AM (ICLR'19)	AR, BS1280	0.00	32.17s	0.25	1.44m	2.44	3.61m
TM† (Arxiv'21)	AR, BS1000	0.09	46.24s	0.02	2.89m	0.46	7.49m
POMO (NeurIPS'20)	AR, G, AUG	0.00	21.71s	0.09	44.84s	0.19	1.26m
Sym-NCO (NeurIPS'22)	AR, G, AUG	0.08	22.44s	0.04	45.97s	0.18	1.79m
Pointerformer (AAAI'23)	AR, G, AUG	0.01	38.58s	0.10	1.35m	0.12	2.29m
ELG† (IJCAI'24)	AR, G, AUG	0.10	55.22s	0.07	2.18m	0.20	4.14m
HierTSP (KDD'24)	AR, G, AUG	0.00	33.48s	0.03	1.92m	0.21	2.79m
GCN (Arxiv'19)	NAR, BS1280	0.80	27.40s	0.53	55.62s	2.53	1.78m
NAR4TSP† (TNNLS'24)	NAR, BS1000	0.29	16.36s	0.25	27.21s	0.81	44.15s
Image Diffusion* (NeurIPS'22)	NAR, G, 2-opt	-	-	1.23	-	2.11	-
DIFUSCO† (NeurIPS'23)	NAR, G, 2-opt	0.14	5.80m	0.10	6.17m	0.23	6.5m
DIFUSCO† (NeurIPS'23)	NAR, S, 2-opt	0.02	1.50h	-0.01	1.21h	-0.01	1.68h
T2TCO† (NeurIPS'23)	NAR, G, 2-opt	0.07	36.01m	0.02	37.61m	0.13	44.51m
T2TCO† (NeurIPS'23)	NAR, S, 2-opt	0.02	1.25h	0.00	1.28h	0.00	1.86h
Blackout DIFUSCO* (Arxiv'25)	NAR, G, 2-opt	-	-	0.06	-	0.61	-
Blackout DIFUSCO* (Arxiv'25)	NAR, S, 2-opt	-	-	**-0.02**	-	0.04	-
EFLOCO (ours)	NAR, G, 2-opt	**0.00**	**5s**	0.07	**8s**	0.38	**12s**
EFLOCO (ours)	NAR, S, 2-opt	**0.00**	1.49m	-0.01	2.00m	**-0.01**	2.95m

Table 1: Performance comparison on TSP datasets. "Gap" indicates the average optimality gap relative to Concorde. "G" and "S" represent Greedy-search and Sampling. Symbol * denotes baseline results from the original paper. Symbol † indicates testing on TSP-20/50 using pre-trained TSP-50 weights provided by the original paper. **Bold** indicates the best solution; underlined indicates the second-best solution.

methods include both AR and NAR models. AR baselines include AM (Kool, van Hoof, and Welling 2019), TM (Bresson and Laurent 2021), POMO (Kwon et al. 2020), Sym-NCO (Kim, Park, and Park 2022), Pointerformer (Jin et al. 2023), ELG (Gao et al. 2024), and HierTSP (Goh et al. 2024). NAR baselines include GCN (Joshi, Laurent, and Bresson 2019), NAR4TSP (Xiao et al. 2024), and diffusion-based models such as Image Diffusion (Graikos et al. 2022), DIFUSCO (Sun and Yang 2023), T2TCO (Li et al. 2023), and Blackout DIFUSCO (Seo 2025).

For ATSP, we benchmark our method against a diverse set of baselines. These include the exact solver CPLEX, classical heuristics (Nearest Neighbor, Nearest Insertion, Furthest Insertion). We also include comparisons with learning-based approaches such as MatNet (Kwon et al. 2021) and the recent diffusion-based IC/DC (Hong et al. 2025).

Main Results

We compare EFLOCO with baselines on the TSP and ATSP datasets and analyze the comparative results as follows.

Results for TSP The results presented in Table 1 demonstrate the strong performance of EFLOCO in both solution quality and inference efficiency. For fair comparisons, we follow DIFUSCO and Image Diffusion by applying greedy-search followed by 2-opt refinement. On TSP-20, EFLOCO achieves a near-zero optimality gap and consistently outperforms diffusion-based baselines such as DIFUSCO and T2TCO on TSP-50 and TSP-100. It is worth noting that while DIFUSCO relies on 50-step denoising and T2TCO relies on 20-step denoising (plus 15-step rewriting for T2TCO), EFLOCO with ATSS achieves comparable or better results with only two inference steps: **yielding orders-of-magnitude faster runtime**. These results validate the effectiveness of EFLOCO in generating high-quality solutions at a minimal computational cost and highlight its potential for deployment in real-world, latency-sensitive scenarios. Furthermore, we compare EFLOCO with diffusion-based baselines under the same inference setting (see Appendix B uploaded to our online code repository).

Results for ATSP As shown by results presented in Table 2, EFLOCO yields superior performance across all problem sizes in terms of both solution quality and inference time. Notably, on ATSP-50, it achieves an average optimality gap of only 0.22% within 59 seconds, significantly outperforming all learning-based baselines. Meanwhile, it also demonstrates competitive performance on ATSP-100, further validating its scalability to larger problem sizes. These results highlight the generality and robustness of our method across different combinatorial optimization settings. Refer to Appendix C uploaded to our online code repository for further experimental results such as time expenditure.

Results for Large-scale TSP We evaluate our method on large-scale TSPs with 500 and 1000 nodes and present the results in Table 3. As shown, EFLOCO scales effectively to larger problem sizes and provides high-quality solutions with reasonable computation cost. Although our greedy de-

Method (venue)	Type	ATSP-20		ATSP-50		ATSP-100	
		Gap(%) ↓	Time ↓	Gap(%) ↓	Time ↓	Gap(%) ↓	Time ↓
Baseline (CPLEX)	Exact	0.00	38.5m	0.00	41.6m	0.00	1.67h
Nearest Neighbor	Heuristic	26.10	-	29.70	-	36.1	-
Nearest Insertion	Heuristic	15.48	-	22.16	-	30.79	-
Furthest Insertion	Heuristic	10.77	-	16.80	-	23.37	-
MatNet (NeurIPS'21)	AR, G	0.49	1.65m	1.13	2.11m	3.67	2.76m
MatNet ×16 (NeurIPS'21)	AR, S	0.09	2.1m	0.24	2.17m	**2.23**	3.3m
IC/DC (Arxiv'24) *	NAR, G	4.80	-	4.27	-	-	-
IC/DC (Arxiv'24) *	NAR, S	0.91	-	1.24	-	-	-
EFLOCO (ours)	NAR, G	0.08	**21s**	0.22	**59s**	3.59	3.8m
EFLOCO (ours)	NAR, S	**0.04**	5.3m	**0.09**	14.5m	**1.82**	59m

Table 2: Performance comparison on ATSP datasets. "Gap" indicates the average optimality gap relative to the CPLEX solution.

Method	TSP-500		TSP-1000	
	Gap(%) ↓	Time ↓	Gap(%) ↓	Time ↓
Concorde	0.00	-	0.00	-
AM	20.99	1.51m	34.75	3.18m
GCN	79.61	6.67m	110.29	28.52m
DIFUSCO (G)	1.75	2.02m	1.96	7.75m
T2TCO (G)	0.91	5.8m	1.36	19.52m
EFLOCO (G)	1.48	**29s**	1.73	**2.01m**
EFLOCO (S)	**0.86**	4.6m	**1.31**	18.91m

Table 3: Performance comparison on large-scale TSPs

Steps	Consistency Loss	Gap (%) ↓
10	Without	0.14
	With	0.05
2	Without	0.19
	With	0.07

Table 4: Ablation study on consistency loss

coding slightly underperforms T2TCO in terms of optimality, our inference is several times faster. When employing sampling, our method achieves even better solution quality while still outperforming T2TCO (greedy) in runtime.

Ablation Studies

We conduct extensive ablation studies to evaluate the design of EFLOCO, focusing on the effectiveness of the proposed 1) velocity-based consistency loss and 2) ATSS strategy.

Velocity-based Consistency Loss The results presented in Table 4 demonstrate that velocity-based consistency loss consistently improves performance in both 10-step and 2-step inference settings. Notably, the performance gain is more pronounced in the 2-step setting. This highlights the importance of enforcing smooth state transitions, particularly in more challenging few-step regimes.

Generality of the Proposed ATSS Strategy We further evaluate the generality of ATSS strategy by applying it to DIFUSCO. Specifically, we retrain DIFUSCO on small-scale datasets using either adaptive or uniform time-step sampling and evaluate each retrained model under three inference schedulers: ATSS, cosine, and linear, under both 10-step and 20-step settings. As show in Table 5, ATSS strategy consistently outperforms standard schedulers across all configurations. These results demonstrate the general applicability of ATSS to diffusion-based combinatorial solvers, leading to improved solution quality.

Steps	Train	Inference	Gap (%) ↓
10	Uniform	Linear	0.5868
	ATSS (ours)	Linear	0.4523
	Uniform	Cosine	0.5471
	ATSS (ours)	Cosine	0.4510
	Uniform	**ATSS (ours)**	0.4743
	ATSS (ours)	**ATSS (ours)**	**0.4352**
20	Uniform	Linear	0.5370
	ATSS (ours)	Linear	0.5201
	Uniform	Cosine	0.5086
	ATSS (ours)	Cosine	0.3968
	Uniform	**ATSS (ours)**	0.4469
	ATSS (ours)	**ATSS (ours)**	**0.3616**

Table 5: Evaluation of training and inference sampling strategies using DIFUSCO re-trained with uniform and adaptive time-step schedules.

Conclusion

We propose EFLOCO, an efficient discrete flow matching method for solving COPs. By learning structure-aware and deterministic solution trajectories, EFLOCO enhances both inference stability and solution quality. A velocity-based consistency loss promotes smooth transitions, while an adaptive time-step scheduling strategy prioritizes more efforts on critical regions. Experiments on both TSPs and ATSPs demonstrate superior performance of EFLOCO in both solution quality and inference efficiency. Going forward, we aim to extend EFLOCO to broader classes of discrete optimization problems and investigate the integration of learned or adaptive path priors for greater generality.

Acknowledgments

This work is supported by the Jilin Provincial Department of Science and Technology Project under Grant 20240101369JC.

References

Applegate, D.; Bixby, R.; Chvatal, V.; and Cook, W. 2006. Concorde TSP Solver. http://www.math.uwaterloo.ca/tsp/concorde.html. Accessed: 2025-07-27.

Applegate, D. L.; Bixby, R. E.; Chvátal, V.; and Cook, W. J. 2007. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press.

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

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

Drakulic, D.; Michel, S.; Mai, F.; Sors, A.; and Andreoli, J.-M. 2023. BQ-NCO: Bisimulation Quotienting for Efficient Neural Combinatorial Optimization. In *Proceedings of Advances in Neural Information Processing Systems*, 77416–77429.

Fu, Z.-H.; Qiu, K.-B.; and Zha, H. 2021. Generalize a Small Pre-trained Model to Arbitrarily Large TSP Instances. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Gao, C.; Shang, H.; Xue, K.; Li, D.; and Qian, C. 2024. Towards Generalizable Neural Solvers for Vehicle Routing Problems via Ensemble with Transferrable Local Policy. In *Proceedings of International Joint Conference on Artificial Intelligence*, 6914–6922.

Gat, I.; Remez, T.; Shaul, N.; Kreuk, F.; Chen, R. T. Q.; Synnaeve, G.; Adi, Y.; and Lipman, Y. 2024. Discrete Flow Matching. In Globerson, A.; Mackey, L.; Belgrave, D.; Fan, A.; Paquet, U.; Tomczak, J.; and Zhang, C., eds., *Advances in Neural Information Processing Systems*, volume 37, 133345–133385.

Goh, Y. L.; Cao, Z.; Ma, Y.; Dong, Y.; Dupty, M. H.; and Lee, W. S. 2024. Hierarchical Neural Constructive Solver for Real-world TSP Scenarios. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 884–895.

Graikos, A.; Malkin, N.; Jojic, N.; and Samaras, D. 2022. Diffusion Models as Plug-and-Play Priors. In *Thirty-Sixth Conference on Neural Information Processing Systems*.

Helsgaun, K. 2017. An extension of the Lin–Kernighan–Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. *Roskilde: Roskilde University*, 24–50.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising Diffusion Probabilistic Models. *arXiv preprint arxiv:2006.11239*.

Hong, S.-H.; Kim, H.-S.; Jang, Z.; Yoon, D.; Song, H.; and Lee, B.-J. 2025. Unsupervised Training of Diffusion Models for Feasible Solution Generation in Neural Combinatorial Optimization. arXiv:2411.00003.

Jin, Y.; Ding, Y.; Pan, X.; He, K.; Zhao, L.; Qin, T.; Song, L.; and Bian, J. 2023. Pointerformer: Deep reinforced multi-pointer transformer for the traveling salesman problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 8132–8140.

Joshi, C. K.; Cappart, Q.; Rousseau, L.-M.; and Laurent, T. 2021. Learning TSP Requires Rethinking Generalization. In *International Conference on Principles and Practice of Constraint Programming*.

Joshi, C. K.; Laurent, T.; and Bresson, X. 2019. An efficient graph convolutional network technique for the travelling salesman problem. In *Proceedings of INFORMS Annual Meeting, Session on Boosting Combinatorial Optimization using Machine Learning*, 1–17.

Kim, G.; Ong, Y.-S.; Heng, C. K.; Tan, P. S.; and Zhang, N. A. 2015. City Vehicle Routing Problem (City VRP): A Review. *IEEE Transactions on Intelligent Transportation Systems*, 16: 1654–1666.

Kim, M.; Park, J.; and Park, J. 2022. Sym-NCO: Leveraging Symmetricity for Neural Combinatorial Optimization. In *Proceedings of Advances in Neural Information Processing Systems*, volume 35, 1936–1949.

Kool, W.; van Hoof, H.; and Welling, M. 2019. Attention, Learn to solve routing problems! In *Proceedings of International Conference on Learning Representations*.

Kwon, Y.-D.; Choo, J.; Kim, B.; Yoon, I.; Gwon, Y.; and Min, S. 2020. POMO: Policy Optimization with Multiple Optima for Reinforcement Learning. In *Proceedings of Advances in Neural Information Processing Systems*, 21188–21198.

Kwon, Y.-D.; Choo, J.; Yoon, I.; Park, M.; Park, D.; and Gwon, Y. 2021. Matrix encoding networks for neural combinatorial optimization. In Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*.

Li, Y.; Guo, J.; Wang, R.; and Yan, J. 2023. T2T: From Distribution Learning in Training to Gradient Search in Testing for Combinatorial Optimization. In *Advances in Neural Information Processing Systems*.

Li, Y.; Guo, J.; Wang, R.; Zha, H.; and Yan, J. 2024. Fast t2t: Optimization consistency speeds up diffusion-based training-to-testing solving for combinatorial optimization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Lin, S.; and Kernighan, B. W. 1973. An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research*, 21: 498–516.

Lipman, Y.; Chen, R. T. Q.; Ben-Hamu, H.; Nickel, M.; and Le, M. 2023. Flow Matching for Generative Modeling. In *The Eleventh International Conference on Learning Representations*.

Liu, X.; Gong, C.; and Liu, Q. 2022. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*.

Luo, F.; Lin, X.; Liu, F.; Zhang, Q.; and Wang, Z. 2023. Neural Combinatorial Optimization with Heavy Decoder:

Toward Large Scale Generalization. In *Proceedings of Advances in Neural Information Processing Systems*, volume 36, 8845–8864.

Luo, F.; Lin, X.; Wang, Z.; Tong, X.; Yuan, M.; and Zhang, Q. 2024. Self-Improved Learning for Scalable Neural Combinatorial Optimization. ArXiv:2403.19561, arXiv:2403.19561.

Seo, J. P. 2025. Blackout DIFUSCO. arXiv:2502.05221.

Shaul, N.; Gat, I.; Havasi, M.; Severo, D.; Sriram, A.; Holderrieth, P.; Karrer, B.; Lipman, Y.; and Chen, R. T. Q. 2025. Flow Matching with General Discrete Paths: A Kinetic-Optimal Perspective. In *The Thirteenth International Conference on Learning Representations*.

Sun, Z.; and Yang, Y. 2023. DIFUSCO: Graph-based Diffusion Solvers for Combinatorial Optimization. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Wang, M.; Zhou, Y.; Cao, Z.; Xiao, Y.; Wu, X.; Pang, W.; Jiang, Y.; Yang, H.; Zhao, P.; and Li, Y. 2025. An Efficient Diffusion-based Non-Autoregressive Solver for Traveling Salesman Problem. In *Proceedings of the 31th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Wu, X.; Wang, D.; Wen, L.; Xiao, Y.; Wu, C.; Wu, Y.; Yu, C.; Maskell, D. L.; and Zhou, Y. 2024. Neural Combinatorial Optimization Algorithms for Solving Vehicle Routing Problems: A Comprehensive Survey with Perspectives. ArXiv:2406.00415, arXiv:2406.00415.

Wu, X.; Wang, D.; Wu, C.; Wen, L.; Miao, C.; Xiao, Y.; and Zhou, Y. 2025. Efficient Heuristics Generation for Solving Combinatorial Optimization Problems Using Large Language Models. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

Xia, Y.; Yang, X.; Liu, Z.; Liu, Z.; Song, L.; and Bian, J. 2024. Position: Rethinking Post-Hoc Search-Based Neural Approaches for Solving Large-Scale Traveling Salesman Problems. In *Proceedings of International Conference on Machine Learning*, 54178–54190.

Xiao, Y.; Wang, D.; Li, B.; Chen, H.; Pang, W.; Wu, X.; Li, H.; Xu, D.; Liang, Y.; and Zhou, Y. 2024. Reinforcement Learning-based Non-Autoregressive Solver for Traveling Salesman Problems. *IEEE Transactions on Neural Networks and Learning Systems*.

Xiao, Y.; Wu, Y.; Cao, R.; Wang, D.; Cao, Z.; Zhao, P.; Li, Y.; Zhou, Y.; and Jiang, Y. 2025. DGL: Dynamic Global-Local Information Aggregation for Scalable VRP Generalization with Self-Improvement Learning. In *Proceedings of International Joint Conference on Artificial Intelligence*, 1–9.

Xue, S.; Liu, Z.; Chen, F.; Zhang, S.; Hu, T.; Xie, E.; and Li, Z. 2024. Accelerating Diffusion Sampling with Optimized Time Steps. *arXiv preprint arXiv:2402.17376*.

Yang, L.; Zhang, Z.; Song, Y.; Hong, S.; Xu, R.; Zhao, Y.; Zhang, W.; Cui, B.; and Yang, M.-H. 2023. Diffusion Models: A Comprehensive Survey of Methods and Applications. *ACM Comput. Surv.*, 56(4).

Yang, L.; Zhang, Z.; Zhang, Z.; Liu, X.; Xu, M.; Zhang, W.; Meng, C.; Ermon, S.; and Cui, B. 2024. Consistency Flow Matching: Defining Straight Flows with Velocity Consistency. *CoRR*, abs/2407.02398.

Zhao, P.; Cao, Z.; Wang, D.; Song, W.; Pang, W.; Zhou, Y.; and Jiang, Y. 2025a. Visual-Enhanced Multimodal Framework for Flexible Job Shop Scheduling Problem. In *Proceedings of the 33rd ACM International Conference on Multimedia*, 2496–2505.

Zhao, P.; Zhou, Y.; Wang, D.; Cao, Z.; Xiao, Y.; Wu, X.; Li, Y.; Liu, H.; Du, W.; Jiang, Y.; et al. 2025b. Dual Operation Aggregation Graph Neural Networks for Solving Flexible Job-Shop Scheduling Problem with Reinforcement Learning. In *Proceedings of the ACM on Web Conference 2025*, 4089–4100.