# DEG: Diffusion-Enhanced GFlowNet for Solving Vehicle Routing Problems

**Ni Zhang**[1], **Zhiqin Zhang**[1,*], **Ling Pan**[2], **Hoong Chuin Lau**[1], **Zhiguang Cao**[1]
[1]Singapore Management University
[2] Hong Kong University of Science and Technology
ni.zhang.2025@phdcs.smu.edu.sg, zhiqinzhang@smu.edu.sg, lingpan@ust.hk
hclau@smu.edu.sg, zgcao@smu.edu.sg

## Abstract

Traditional neural solvers for solving vehicle routing problems (VRPs) often suffer from limited solution diversity, motivating the development of Generative Flow Network (GFlowNet)–based models. However, the effectiveness of these models is frequently constrained by insufficient flow expansion in high-reward regions, limiting their ability to distribute probability across promising solution routes. The deeper exploration could yield superior results. Diffusion models, in contrast, provide stronger structural guidance for exploration. These two paradigms are naturally complementary: GFlowNet can supply edge-level signals for diffusion to embed, while diffusion can guide broader exploration. Leveraging this synergy, we propose Diffusion-Enhanced GFlowNet (DEG), a novel framework that integrates GFlowNet with a diffusion model to encourage richer flow expansion toward high-reward regions and derive higher-quality solutions. Specifically, DEG exploits GFlowNet's inherent diversity to generate edge-specific backward signals, applies the stochastic noise schedule of diffusion to perturb these signals, and then denoises them within the GFlowNet paradigm. To further improve scalability, we introduce a specialized decoder capable of dynamically adapting to diverse problem scales. Extensive experimental evaluations on synthetic and real-world datasets, including instances with up to 10,000 nodes, demonstrate that DEG consistently achieves favorable performance compared to baseline methods. The code is available at https://github.com/ZHANG-NI/DEG.

## 1 Introduction

The Vehicle Routing Problem (VRP) is a fundamental NP-hard combinatorial optimization problem that generalizes the Traveling Salesman Problem (TSP) by incorporating practical constraints such as vehicle capacity limitations and route feasibility requirements. The objective of VRPs is to minimize overall travel costs by determining optimal routes for a fleet of vehicles dispatched from a central depot to serve multiple customers while satisfying these constraints. VRPs are central to a wide range of applications, including logistics and distribution planning [6, 7, 11, 22, 29, 33, 67], agricultural field operations [57, 60, 55], and urban delivery systems [10, 17, 14]. Owing to their importance, VRPs have attracted extensive research attention. Traditional metaheuristics [8, 2, 40, 44, 13, 31, 48, 41, 3] have demonstrated effectiveness in solving VRPs. However, they are often computationally expensive and heavily dependent on carefully designed solution components and extensive hyperparameter tuning tailored to specific instances. These limitations have spurred growing interest in deep learning–based neural solvers, which aim to provide data-driven heuristics that improve inference efficiency while reducing reliance on expert knowledge.

*Corresponding author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Recent neural solvers for VRPs can be broadly categorized into Autoregressive (AR) and Non-Autoregressive (NAR) models. AR models, such as those based on pointer networks and Transformer architectures [32, 38, 9, 24, 62], have achieved strong solution quality. However, their sequential decoding process introduces significant computational bottlenecks, limiting scalability to large instances. NAR models provide a promising alternative by generating complete solutions in a single forward pass, greatly improving inference efficiency. Early NAR methods, however, struggled with solution quality, motivating the development of Generative Flow Network (GFlowNet) [4, 5]. While GFlowNet-based solvers [68, 27, 66] leverage sampling diversity to improve solution quality, they often converge to sub-optimal route subsets and fail to fully explore high-reward regions, leading to insufficient flow expansion into these more promising areas and hindering the discovery of higher-quality solutions.

Diffusion models [52, 21] recently emerged as a powerful approach in the VRP domain for their strong exploration capability [54, 34, 35, 56]. Yet existing diffusion-based methods remain largely confined to the TSP and depend on supervised frameworks to generate edge-level noise signals. While GFlowNet and diffusion models each have distinct strengths and limitations, their integration for VRP solving has not been explored. These two paradigms are naturally complementary: diffusion can guide GFlowNet toward more effective flow expansion in high-reward regions, and GFlowNet can provide edge-level signals and extend the applicability of diffusion methods.

Based on this, we propose Diffusion-Enhanced GFlowNet (DEG), a novel framework that integrates GFlowNet with diffusion models for solving VRPs. DEG leverages the intrinsic diversity of GFlowNet to generate edge-specific backward signals, which serve as a noise basis for the diffusion process. Unlike conventional trajectory-level backward signals that collapse into a single aggregated value, edge-specific signals provide richer structure for diffusion noise modeling. DEG perturbs these signals with a stochastic noise schedule and denoises them within the GFlowNet paradigm, preventing premature convergence to sub-optimal policies and promoting broader exploration of high-reward regions. This mechanism enhances forward probability allocation across promising routes and enables more effective flow expansion, enabling the discovery of higher-quality solutions. To further enhance generalization across varying problem scales, we develop the Graph-Scale Adapter (GSA), a specialized module that aligns spatial statistics across different graph scales and significantly improves scalability. Experimental evaluations demonstrate DEG's strong performance and scalability on both TSP and Capacitated Vehicle Routing Problem (CVRP), achieving competitive results compared to existing baselines and exhibiting robustness on instances containing up to 10,000 nodes. Our contributions are summarized as follows:

- We propose Diffusion-Enhanced GFlowNet (DEG), which leverages intrinsic diversity of GFlowNet to generate edge-specific backward signals that serve as the adaptive noise basis for integrating diffusion, then applies the diffusion's noise schedule to inject noise into these signals, and denoises them through the GFlowNet paradigm, enabling deeper flow expansion toward high-reward regions to derive better solutions.
- We design the Graph-Scale Adapter (GSA) to align spatial statistics across varying instance sizes, which enhances the model's ability to generalize to large-scale problems.
- We conduct comprehensive experiments on TSP and CVRP, and results show that DEG consistently achieves favorable results on both synthetic and real-world datasets.

## 2 Related Work

The deep learning based neural solvers for VRPs can be broadly classified into Autoregressive (AR) and Non-Autoregressive (NAR) methods, respectively.

**AR methods** sequentially construct solutions step-by-step, conditioned on previously selected nodes. Transformer-based architectures have been particularly influential due to their ability to effectively capture long-range dependencies. A notable early example is the Attention Model (AM) [30], which applies self-attention mechanisms to enable end-to-end routing policy learning. To further enhance solution diversity and robustness, POMO [32] builds on AM by employing multi-start policy optimization, exploring multiple potential optimal paths simultaneously. Building on the strengths of autoregressive architectures, LEHD [38] introduces a light-encoder, heavy-decoder design, achieving substantial performance gains across standard VRP benchmarks.

**NAR methods**, by contrast, generate complete solutions in a single forward pass, drastically reducing inference time and enabling parallel solution generation. This characteristic makes them particularly appealing for large-scale applications. Diffusion-based models represent a recent and promising class of NAR approaches. DIFUSCO [54] is a pioneering diffusion-based model tailored for the TSP variant of VRPs. Building upon this foundation, T2T [34], Fast T2T [35] and DEITSP [56] significantly improve solution quality while retaining inference efficiency. However, all these diffusion-based methods rely on supervised learning with optimal solutions and are mainly limited to TSP. GFlowNet-based solvers form another mainstream approach. GFACS [27] employs GFlowNet to enhance the ACO heuristic. AGFN [68] adopts an adversarial strategy to generate routes in an end-to-end manner. Different from those methods, in this work, we propose Diffusion-Enhanced GFlowNet (DEG), which leverages diffusion to enhance the exploration of high-reward regions, thereby improving the solution quality for VRPs.

## 3 Preliminary

**Problems Definition.** In both the TSP and CVRP, the instance is defined on a complete graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{n_0, n_1, \ldots, n_v\}$ denotes the set of nodes and $\mathcal{E}$ the set of edges. The cost $d(n_i, n_j)$ associated with each edge is typically defined as the Euclidean distance between nodes $n_i$ and $n_j$. In the TSP, the objective is to find a minimum-cost Hamiltonian circuit that visits all nodes exactly once. In the CVRP, $n_0$ represents the depot, and the remaining nodes correspond to customers with non-negative demands $d_i$. The goal is to construct a set of depot-rooted routes $\tau = \{q_1, \ldots, q_N\}$ such that each customer is visited exactly once, the total demand on each route does not exceed the vehicle capacity $C$, and the overall travel cost is minimized.

**Diffusion Models.** Diffusion models have recently emerged as powerful generative approaches, achieving favorable results in domains such as image synthesis [52, 21, 49, 15], audio generation [23, 28], and drug design [51, 58]. Diffusion models operate by corrupting clean inputs with noise and denoise them to reconstruct the original data. Typically, given a clean input $x_0$, the noise process perturbs it by adding Gaussian noise to obtain a noisy sample $x_t$ at time step $t$, following

$$q(x_t \mid x_0) = \mathcal{N}(x_t; \sqrt{\alpha_t} x_0, (1 - \alpha_t)\mathbf{I}), \tag{1}$$

where $\alpha_t \in (0, 1)$ controls the noise level. The model can capture the structure of the target distribution by extracting meaningful signals from noisy states and recovering the original data.

**GFlowNet.** Generative Flow Network (GFlowNet) is a variational inference algorithm that treat sampling from a target probability distribution as a sequential decision-making process [4, 5], and is widely applied to structured generation tasks, including molecule design [43, 26, 69], neural architecture search [53, 25], and combinatorial optimization [65, 63, 12]. The objective of GFlowNet is to learn a policy that samples objects in proportion to a given reward distribution. From a high-level perspective, a GFlowNet consists of two complementary components: forward sampling policy and backward policy. The forward pass includes the source flow $Z(\mathcal{G}; \theta)$ and a forward policy $P_F(\tau; \theta)$ that governs trajectory construction from initial to terminal states. The backward pass includes the reward function $R(\tau)$ and a backward policy $P_B(\tau)$ that defines reverse transitions. Together, these bidirectional passes form the basis for training via trajectory balance objectives [39], defined as:

$$\mathcal{L}_{\text{TB}}(\tau; \theta) = \left( \log \frac{Z(\mathcal{G}; \theta) P_F(\tau; \boldsymbol{\theta})}{R(\tau) P_B(\tau)} \right)^2. \tag{2}$$

## 4 Diffusion-Enhanced GFlowNet

We introduce Diffusion-Enhanced GFlowNet (DEG), a framework designed to overcome a core limitation of existing GFlowNet-based VRP solvers, namely, their inability to effectively expand flow toward high-reward regions. DEG integrates GFlowNet with diffusion by leveraging edge-specific backward signals, which serve as a structured noise basis for the diffusion process. A stochastic noise schedule perturbs these signals, which are subsequently denoised within the GFlowNet paradigm, enabling richer exploration and more sufficient flow expansion. To further strengthen scalability and inference quality across varying problem sizes, DEG incorporates a Graph-Scale Adapter (GSA) decoder, which dynamically aligns spatial statistics across different graph scales.

### 4.1 Edge-specific Backward Signals and Noise Schedule

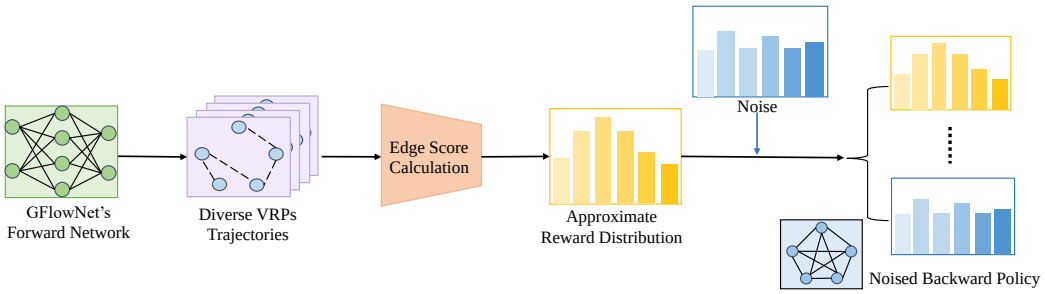



Figure 1: Illustration of Edge-specific Backward Signals Generation and Noise Schedule.

Figure 1 illustrates the generation of edge-specific backward signals and noise schedule. We first use GFlowNet to generate a diverse set of trajectories and assign standardized scores to each edge, estimating an approximate reward distribution that serves as edge-specific backward signals. Although the individual trajectories produced during GFlowNet sampling may not be optimal, their collective information forms a robust reward distribution that guides the diffusion learning process. A noise schedule then injects controlled noise into this distribution to derive a noised backward policy.

By leveraging the inherent diversity of GFlowNet, the model can generate a wide range of trajectories, after which the edge score calculation is applied. The edge score calculation begins by normalizing the reward of each trajectory $\tau_k \in \mathcal{T}$ using the following formulation:

$$-\log R(\tau_k) = D(\tau_k) - \frac{1}{m}\sum_{l=1}^{m} D(\tau_l), \tag{3}$$

where $D(\tau_k)$ is the total distance of trajectory $\tau_k$, defined as: $D(\tau_k) = \sum_{l=1}^{N-1} d(q_l, q_{l+1})$. Here, $d(q_l, q_{l+1})$ denotes the Euclidean distance between nodes $q_l$ and $q_{l+1}$. A higher reward $R(\tau_k)$ corresponds to a shorter, higher-quality trajectory. We define the reward of an edge $e$ as the average reward of the trajectories in which it appears:

$$R(e) = \frac{1}{|\{\tau_k \mid \tau_k \in \mathcal{T},\ e \in \tau_k\}|} \sum_{\tau_k \in \mathcal{T},\, e \in \tau_k} R(\tau_k), \tag{4}$$

where the denominator counts the number of trajectories containing edge $e$. This edge-level reward $R(e)$ reflects the global significance of edge $e$ across the sampled solution space. To standardize $R(e)$ across different instances, we use z-score normalization [42]: $\hat{R}(e) = \frac{R(e)-\mu_R}{\sigma_R}$, where $\mu_R$ and $\sigma_R$ denote the mean and standard deviation of edge rewards within the instance. To convert the standardized scores into bounded confidence values, we apply the cumulative distribution function (CDF) [50] of the standard normal distribution:

$$x_0 = \Phi\left(\hat{R}(e)\right), \tag{5}$$

where $\Phi(\cdot)$ denotes the CDF of the standard normal distribution. The value $x_0$ represents an approximate reward distribution that provides edge-specific backward signals and serves as the basis for adding noise to integrate diffusion into GFlowNet. The noise schedule then injects Gaussian noise at each training step to obtain a noised backward policy $x_t$:

$$x_t = \sqrt{\alpha_t} \cdot x_0 + \sqrt{1-\alpha_t} \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0,1), \tag{6}$$

where $\alpha_t = \prod_{k=1}^{t}(1-\beta_k)$, and $\beta_k \in (0,1)$ is the corruption ratio at diffusion step $k$. To ensure full transformation of the clean signal into noise by the final training iteration $I$, $\beta_k$ should be defined such that $\prod_{k=1}^{I}(1-\beta_k) \approx 0$. We use continuous Gaussian noise to match the inherently continuous nature of the backward policy in GFlowNet.

**Effect of Edge-specific Backward Signals and Noise Schedule on High-Reward Region Exploration**. DEG uses a noise schedule that progressively corrupts the edge-specific backward signals

during training, together with the Trajectory Balance (TB) objective [39]. We characterize the learning target induced by this schedule at each noise level, with proofs provided in Appendix B. In practice, the noised signal is clamped as $\tilde{x}_t[e] = \max(\text{clamp}(x_t[e], 0, 1), \delta) \in [\delta, 1]$ with $\delta = 10^{-10}$, ensuring that the backward-side trajectory weight $W_t(\tau) = \prod_{e\in\tau} \tilde{x}_t[e]$, which replaces $P_B(\tau)$ in $\mathcal{L}_{\text{TB}}$ (Eq. 2), remains strictly positive.

Because TB minimizes a squared log-ratio, the injected noise affects the optimum through $\mathbb{E}_\epsilon[\log W_t(\tau)]$. For a fixed graph $\mathcal{G}$ and signal $x_0$, the population minimizer at noise level $t$ is

$$P_F^t(\tau) \propto R(\tau)e^{\mu_t(\tau)}, \qquad \mu_t(\tau) = \sum_{e\in\tau} \mathbb{E}_\epsilon\big[\log \tilde{x}_t[e]\big], \tag{7}$$

which holds exactly at every noise level (Proposition 1). The target can therefore be viewed as reward-proportional sampling modified by an edge-specific factor, whose dependence on $x_0$ is gradually reduced by the noise schedule. At the beginning of training, no noise is added ($\bar{\alpha}_t = 1$), giving

$$P_F^0(\tau) \propto R(\tau) \prod_{e\in\tau} x_0[e]. \tag{8}$$

Compared with $\pi_R \propto R$, this distribution gives more weight to trajectories containing high-scoring edges. Its expected reward is higher than that of $\pi_R$ whenever $\prod_{e\in\tau} x_0[e]$ is positively correlated with $R(\tau)$ (Proposition 2), which motivates constructing $x_0$ from edge-reward statistics. This early stage therefore favors exploitation. Only when $\prod_{e\in\tau} x_0[e] \propto R(\tau)$ does Eq. 8 reduce to $P_F \propto R(\tau)^2$; our CDF-based construction of $x_0$ does not assume this special case.

As the noise increases, the influence of $x_0$ weakens. At full corruption ($\bar{\alpha}_t = 0$), $\tilde{x}_t[e]$ is independent of $x_0$, and $\mu_t(\tau) = c_T|\tau|$ for a constant $c_T$. For TSP, all tours contain exactly $n$ edges, so this term is identical across trajectories and

$$P_F(\tau) \propto R(\tau). \tag{9}$$

Before reaching this endpoint, $\text{KL}(P_F^t \,\|\, \pi_R)$ admits a bound that decreases to zero as $\bar{\alpha}_t \to 0$ (Theorem 1), providing a useful characterization of the late training stage. For CVRP, a trajectory with $K(\tau)$ routes contains $n+K(\tau)$ edges. The endpoint target therefore becomes $P_F(\tau) \propto R(\tau)e^{c_T K(\tau)}$ with $c_T < 0$. It is reward-proportional among solutions with the same number of routes, while additionally favoring solutions with fewer routes (Remark 2).

Overall, the noise schedule gradually removes the edge-specific factor from the learning target, moving from the edge-guided distribution in Eq. 8 toward the reward-proportional distribution in Eq. 9. This gives a natural transition from early exploitation to broader exploration.

### 4.2 Diffusion-enhanced GFlowNet Framework

In our framework, the model learns to construct solutions by drawing on both the noisy backward policy and the reward within the GFlowNet paradigm, which denoises the noised backward policy during training by extracting its informative weights, and using them to enhance forward-policy learning. The loss over a set of trajectories $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_m\}$, based on the denoised forward policy which estimates reward distribution, is defined as:

$$\mathcal{L}(\mathcal{T}; \theta) = \sum_{k=1}^{m} \left( \log \frac{Z(\mathcal{G}; \theta) P_F(\tau_k; \boldsymbol{\theta})}{R(\tau_k) P_B(\tau_k)} \right)^2. \tag{10}$$

Here, source flow $Z(\mathcal{G}; \theta)$ is a learnable partition function derived from Eq. 15, and reward $R(\tau_k)$ is defined in Eq. 3. The forward probability $P_F(\tau_k; \boldsymbol{\theta})$ is computed as:

$$P_F(\tau_k; \boldsymbol{\theta}) = \prod_{t=1}^{N} P_F(s_t | s_{t-1}; \boldsymbol{\theta}), \tag{11}$$

where $s_t = \{q_1, q_2, \ldots, q_t\}$ denotes the state at step $t$ during the construction (or destruction) of the trajectory. The term $P_F(s_t|s_{t-1})$ is obtained from the forward policy $\eta(\mathcal{G}^*, \boldsymbol{\theta})$, which is derived from the GFlowNet forward network described in Sec. A.1.

Moreover, $\eta(\mathcal{G}^*, \boldsymbol{\theta})$ can be viewed as a *heatmap* over all edges of the current graph: it is a $|V| \times |V|$ probability matrix where each entry $\eta(i, j)$ gives the likelihood of selecting the edge $(v_i, v_j)$ from

the current node $v_i$ to a candidate node $v_j$. During decoding, the next node of the route is sampled according to these edge probabilities. After each selection, the graph state $\mathcal{G}^*$ is updated by marking the chosen node as visited and updating feasibility masks (e.g., vehicle capacity), and the process is repeated until all customers are visited and a complete route is constructed.

Similarly, the backward transition probability $P_B(\tau_k)$ in Eq. 10 is computed as:

$$P_B(\tau_k; \boldsymbol{\theta}) = \prod_{k=1}^{N} P_B(s_{k-1}|s_k; \boldsymbol{\theta}), \tag{12}$$

where $P_B(s_{t-1} \mid s_t; \boldsymbol{\theta})$ is derived from the noised backward policy $x_t$. Specifically, for a partial route state $s_t = \{q_1, q_2, \ldots, q_t\}$, this corresponds to removing the last node $q_t$ from the current partial route, which eliminates the edge $(q_t, q_{t-1})$. Thus $P_B(s_{t-1} \mid s_t; \boldsymbol{\theta})$ is taken directly from the value of $x_t$ at the position $(q_t, q_{t-1})$.

### 4.3 Graph-Scale Adapter for Inference

During training, we adopt GFlowNet's inherent sampling strategy to generate diverse trajectories to provide edge-specific signal. This enables efficient sampling and strengthens training effectiveness, with a dedicated model trained for a specific problem size. However, this strategy does not generalize well at inference time due to scale mismatches between training and test instances.

Specifically, the DEG forward policy learns an approximate reward distribution guided by feedback from the noisy backward flow. However, this design may increase the risk of overfitting to spatial patterns specific to the training scale, such as average neighbor distances and node observation radii defined by the sparsity factor $w$. For example, when trained on 100-node instances, the model implicitly learns to solve instances with an average neighbor distance of approximately $\frac{1}{\sqrt{100}}$ and a node observation distance of $\frac{1}{\sqrt{w}}$. When evaluated on a 500-node instance, however, the average neighbor distance drops to $\frac{1}{\sqrt{500}}$, while the observation distance becomes $\frac{1}{\sqrt{5w}}$. This spatial inconsistency hampers generalization and often leads to suboptimal inference. The best outcomes of large-scale instances appear at earlier training stages, when the model has not yet overfitted and its estimation of spatial structures remains effective and less biased. We empirically validate this behavior, with details provided in Appendix C.1.

To mitigate this issue, we propose the *Graph-Scale Adapter* (GSA), which aligns the spatial patterns of test instances to those of fixed-size training instances. GSA introduces a scaling factor $\alpha$ that adjusts node distances and the sparsity factor in the test graph to match the spatial characteristics of the training graph. The scaling factor is defined as: $\alpha \propto \sqrt{\frac{|\mathcal{V}_{\text{test}}|}{|\mathcal{V}_{\text{train}}|}}$. This adjustment ensures structural consistency across different scales. For instance, if the model is trained on 100-node graphs and evaluated on 500-node graphs, applying GSA ensures that the average neighbor distances and node observation radius remain consistent, approximately $\frac{1}{\sqrt{100}}$ and $\frac{1}{\sqrt{w}}$. To assess the effectiveness of GSA, we apply it during inference on instances of varying sizes. As shown in Appendix C.1, GSA effectively alleviates spatial mismatches caused by scale-specific overfitting and significantly improves the model's ability to generalize across a broad range of instance sizes.

## 5 Experiments

We present experimental results to evaluate the performance of DEG. We first assess its effectiveness on both synthetic instances for TSP and CVRP following previous works [27, 68]. Since our model integrates GFlowNet with diffusion, we primarily compare it against GFlowNet-based and diffusion-based baselines. We then evaluate the model's generalization ability on public benchmarks, including TSPLib [1] and CVRPLib [47]. Finally, we conduct ablation studies to examine the contribution of individual components in DEG. We additionally conduct more experiments on multiple scale training, GSA applicability, diversity test, learning efficiency and applicability to other COPs, the details of which are presented in Appendix C.4 to C.7

Following prior work on VRP [54, 27, 68, 64, 56], we use synthetic datasets for training both TSP and CVRP models. Each TSP instance consists of nodes represented only by their 2D coordinates, sampled independently from the unit square $[0, 1]^2$, without any depot or demand. The objective is

Table 1: Comparison of performance and runtime on TSP with varying node sizes. Obj. denotes the total route length; Gap (%) is computed relative to LKH; and Time (s) indicates the average time required to solve a single instance. For all metrics, lower values indicate better results.

| Method | 500 | | | 1000 | | | 3000 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Obj. | Time(s) | Gap(%) | Obj. | Time(s) | Gap(%) | Obj. | Time(s) | Gap(%) |
| LKH | 16.55 | 31.80 | – | 23.14 | 97.62 | – | 40.31 | 776.23 | – |
| GFACS (AISTATS'25) | 22.97 | 8.43 | 38.79 | 38.58 | 18.90 | 66.72 | 69.63 | 81.15 | 72.74 |
| AGFN (ICLR'25) | 19.04 | 0.25 | 15.05 | 27.10 | 0.68 | 17.11 | 55.37 | 2.91 | 37.36 |
| DIFUSCO (NeurIPS'23) | 19.12 | 3.52 | 15.17 | 27.70 | 13.60 | 19.71 | 50.39 | 125.02 | 25.01 |
| T2T (NeurIPS'23) | 18.91 | 3.18 | 14.26 | 27.31 | 12.95 | 18.02 | 56.99 | 115.71 | 41.38 |
| Fast T2T (NeurIPS'24) | 18.76 | 0.37 | 13.35 | 27.25 | 1.54 | 17.76 | 57.73 | 4.76 | 43.22 |
| DEITSP (KDD'25) | 18.64 | 0.42 | 12.63 | 27.11 | 1.68 | 17.16 | 54.37 | 5.35 | 34.81 |
| DEG | **18.47** | 0.28 | **11.60** | **26.46** | 0.72 | **14.35** | **47.82** | 3.06 | **18.63** |

| Method | 5000 | | | 8000 | | | 10000 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Obj. | Time(s) | Gap(%) | Obj. | Time(s) | Gap(%) | Obj. | Time(s) | Gap(%) |
| LKH | 51.84 | 1608.11 | – | 65.89 | 4123.37 | – | 75.78 | 5350.91 | – |
| GFACS (AISTATS'25) | 97.93 | 143.19 | 88.91 | 120.76 | 275.22 | 83.28 | 140.84 | 326.04 | 85.85 |
| AGFN (ICLR'25) | 80.93 | 6.65 | 56.12 | 118.53 | 13.09 | 79.89 | 143.85 | 14.07 | 89.83 |
| DIFUSCO (NeurIPS'23) | – | – | – | – | – | – | – | – | – |
| T2T (NeurIPS'23) | – | – | – | – | – | – | – | – | – |
| Fast T2T (NeurIPS'24) | – | – | – | – | – | – | – | – | – |
| DEITSP (KDD'25) | – | – | – | – | – | – | – | – | – |
| DEG | **63.64** | 6.82 | **22.57** | **83.13** | 14.97 | **26.16** | **94.67** | 16.47 | **24.94** |

*"–" indicates that the model was unable to produce a result for a given instance either within the 90-minute time limit or due to GPU memory constraints.*

to compute the shortest tour that visits each node exactly once and returns to the start. For CVRP, each instance includes a single depot and multiple customer nodes served by a vehicle with a fixed capacity $C = 50$. Node locations (both depot and customers) are uniformly sampled from $[0, 1]^2$. Each customer is assigned a demand independently drawn from a uniform distribution $U[a, b]$, where $a = 1$ and $b = 9$. For training, DEG is trained on 100-node instances for both TSP and CVRP. For evaluation, we generate synthetic test sets containing 128 instances per problem size, ranging from 500 to 10,000 nodes (500, 1,000, 3,000, 5,000, 8,000 and 10,000).

Following standard practice [54, 56, 18], we adopt a linear noise schedule with $\beta_1 = 10^{-4}$ and $\beta_I = 0.02$. The model is trained for $I = 6000$ steps, using a batch size of 10 instances per step. We use the AdamW optimizer [36] with an initial learning rate of $10^{-4}$, decayed to $10^{-5}$ via a CosineAnnealingLR schedule [37]. All experiments are conducted on a system with an NVIDIA H100 NVL GPU and an AMD EPYC 9554 CPU.

### 5.1 Comparison Study on TSP

As prior diffusion-based work has primarily focused on the TSP [54, 34, 35, 56] in the VRP domain, we first evaluate DEG on synthetic TSP instances. Besides the heuristic solver LKH [20], we mainly compare it against both GFlowNet-based solvers including GFACS [27] and AGFN [68], and diffusion-based solvers including DIFUSCO [54], T2T [34], Fast T2T [35], and DEITSP [56]. All models are trained on 100-node instances. To ensure fair comparison, and due to space constraints, Table 1 reports the performance of models *without* 2-opt post-processing. Following [54, 34, 35, 56], results with 2-opt post-processing are provided in Appendix C.2 and Table 6 for completeness. We also provide the comparison on small-scale instances in Appendix C.3 and Table 7.

As shown in Table 1, DEG outperforms all GFlowNet-based and diffusion-based solvers in terms of solution quality. Specially, it achieves improvements of up to $67.43\%$, $34.19\%$, $5.10\%$, $16.10\%$, $17.17\%$, and $12.05\%$ over GFACS, AGFN, DIFUSCO, T2T, Fast T2T, and DEITSP, respectively. Notably, the performance gap between our model and other baselines becomes more significant as the instance size increases.

### 5.2 Comparison Study on CVRP

We present results on the CVRP instances, focusing primarily on comparisons with GFlowNet-based solvers, as no existing diffusion-based model is capable of solving CVRP. Besides the heuristic solver LKH [20], we compare DEG against GFlowNet-based solvers including GFACS [27] and AGFN

Table 2: Comparison of performance and runtime on CVRP with varying node sizes. Obj. denotes the total route length; Gap (%) is computed relative to LKH; and Time (s) indicates the average time required to solve a single instance. For all metrics, lower values indicate better results.

| Method | 500 | | | 1000 | | | 3000 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Obj. | Time(s) | Gap(%) | Obj. | Time(s) | Gap(%) | Obj. | Time(s) | Gap(%) |
| LKH | 63.14 | 134.40 | – | 119.52 | 638.40 | – | 380.26* | 671.68 | – |
| GFACS (AISTATS'25) | 80.77 | 10.43 | 27.92 | 158.97 | 22.07 | 33.01 | 393.27 | 130.81 | 3.42 |
| AGFN (ICLR'25) | 71.05 | 0.43 | 12.53 | 133.97 | 0.75 | 12.09 | 368.97 | 3.84 | -2.97 |
| LEHD (NeurIPS'23) | **64.68** | 0.52 | **2.44** | **122.85** | 10.15 | **2.79** | **355.95** | 33.75 | **-6.39** |
| POMO (NeurIPS'20) | 80.21 | 0.50 | 27.04 | 164.17 | 0.92 | 37.36 | 1094.48 | 7.01 | 187.82 |
| POMO(*8) (NeurIPS'20) | 75.45 | 0.67 | 19.50 | 135.74 | 3.13 | 12.74 | 704.99 | 18.26 | 85.40 |
| DEG | 69.76 | 0.45 | 10.48 | 129.67 | 0.80 | 8.49 | 362.11 | 4.08 | -4.77 |

| Method | 5000 | | | 8000 | | | 10000 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Obj. | Time(s) | Gap(%) | Obj. | Time(s) | Gap(%) | Obj. | Time(s) | Gap(%) |
| LKH(1000) | 642.95* | 2263.34 | – | – | – | – | – | – | – |
| GFACS (AISTATS'25) | 650.62 | 262.63 | 1.19 | 1038.30 | 529.04 | – | 1439.19 | 607.26 | – |
| AGFN (ICLR'25) | 608.58 | 7.42 | -5.35 | 917.24 | 15.24 | – | 1140.46 | 18.58 | – |
| LEHD (NeurIPS'23) | 603.89 | 96.75 | -6.08 | 938.06 | 336.70 | – | 1213.13 | 556.88 | – |
| POMO (NeurIPS'20) | 1438.97 | 19.32 | 123.81 | – | – | – | – | – | – |
| POMO(*8) (NeurIPS'20) | 1381.63 | 41.86 | 114.89 | – | – | – | – | – | – |
| DEG | **596.43** | 10.22 | **-7.24** | **907.65** | 20.32 | **–** | **1125.04** | 24.95 | **–** |

*"–" indicates that the model was unable to produce a result for a given instance either within the 90-minute time limit or due to GPU memory constraints. * denotes LKH run with 1,000 iterations due to time limits.*

Table 3: Comparison of performance on TSPLib. Gap (%) is computed relative to existing best-known solutions, and lower values indicate better results.

| TSP | GFACS | AGFN | DEG | DIFUSCO* | T2T* | Fast T2T* | DEITSP* | DEG* |
|---|---|---|---|---|---|---|---|---|
| 50-200 | 29.60 | 19.41 | **14.37** | 0.96 | 0.37 | 0.26 | 0.81 | **0.21** |
| 200-1000 | 51.67 | 25.51 | **17.68** | 3.45 | 1.84 | 1.26 | 1.46 | **1.03** |

*"–" indicates that the model was unable to produce a result for a given instance either within the 90-minute time limit or due to GPU memory constraints. * represents using two-opt after trajectories are generated.*

[68], as well as other neural solvers such as LEHD [38] and POMO [32] to ensure comprehensive evaluation. For LKH, we use a setting of 1000 iterations on 3000-, 5000-, 8000- and 10000-node instances, as larger-scale problems require substantial computational time. Also, all neural models are trained on 100-node synthetic instances, and the routes are evaluated directly without post-processing. Experimental results on small-scale instances are further provided in Appendix C.3, and the corresponding comparisons are summarized in Table 7.

As shown in Table 2, DEG achieves better results than GFACS by $13.63\%$, $18.43\%$, $7.92\%$, $8.33\%$, $12.58\%$ and $21.83\%$ on the 500-, 1000-, 3000-, 5000-, 8000- and 10000-node instances, and outperforms AGFN by $1.82\%$, $3.21\%$, $1.86\%$, $2.00\%$, $1.05\%$ and $1.35\%$ on the same instance sizes. When compared to Transformer-based AR solvers, NAR methods often struggle to match their performance without the aid of improvement techniques, partly due to inherent limitations in network depth and complexity. Without relying on additional post-processing techniques, our model DEG is able to deliver competitive performance on large-scale instances, while also offering notable advantages in inference efficiency. Compared to state-of-the-art AR mode, LEHD, which adopts a heavy decoder transformer network, our model reduces the objective value by $1.24\%$, $3.28\%$ and $7.26\%$ on the 5000-, 8000- and 10000-node instances, respectively, and reduces inference time by $13.46\%$, $92.61\%$, $87.91\%$, $87.37\%$, $93.96\%$ and $95.58\%$ on 500-, 1000-, 3000-, 8000- and 5000-node instances. Furthermore, our model consistently outperforms POMO in objective value and achieves shorter inference time across all instance sizes. Overall, our model achieves competitive results against GFlowNet-based solvers and other neural models, and offers significantly shorter inference time, particularly on large-scale instances. Notably, DEG achieves better performance than both LEHD and POMO on real-world datasets, as detailed in Sec. 5.3 and Table 4.

Table 4: Comparison of performance on CVRPLib. Gap (%) is computed relative to existing best-known solutions, and lower values indicate better results.

| Gap(%) | L1 (3k) | L2 (4k) | A1 (6k) | A2 (7k) | G1 (10k) | G2 (11k) | B1 (15k) | B2 (16k) |
|---|---|---|---|---|---|---|---|---|
| GFACS | 120.21 | 1800.04 | 114.37 | 238.19 | 278.11 | 2242.09 | 348.54 | 2634.74 |
| AGFN | 1144.20 | 27.88 | 29.64 | 24.58 | 142.65 | 27.21 | 20.63 | 160.24 |
| LEHD | **14.04** | 26.30 | 18.90 | 26.40 | 27.23 | 38.45 | 35.94 | 40.76 |
| POMO | 75.30 | 78.16 | 112.27 | 159.22 | – | – | – | – |
| DEG | 19.64 | **26.08** | **14.95** | **23.80** | **23.18** | **24.41** | **17.04** | **23.47** |

Table 5: Ablation Study on TSP and CVRP. Obj. denotes the total route length, and Gap (%) is computed relative to LKH; lower values indicate better results.

| TSP | 500 | | 1000 | | 3000 | | 5000 | | 8000 | | 10000 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Obj. | Gap(%) | Obj. | Gap(%) | Obj. | Gap(%) | Obj. | Gap(%) | Obj. | Gap(%) | Obj. | Gap(%) |
| Backbone | 21.96 | 32.69 | 35.41 | 53.03 | 64.82 | 60.80 | 91.66 | 76.81 | – | – | – | – |
| + Diffusion | 18.54 | 12.02 | 26.74 | 15.56 | 48.45 | 20.19 | 64.97 | 25.33 | – | – | – | – |
| + GSA | **18.47** | **11.60** | **26.46** | **14.35** | **47.82** | **18.63** | **63.64** | **22.57** | **83.13** | **26.16** | **94.67** | **24.94** |

| CVRP | 500 | | 1000 | | 3000 | | 5000 | | 8000 | | 10000 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Obj. | Gap(%) | Obj. | Gap(%) | Obj. | Gap(%) | Obj. | Gap(%) | Obj. | Gap(%) | Obj. | Gap(%) |
| Backbone | 76.88 | 21.76 | 148.61 | 24.34 | 389.18 | 2.35 | – | – | – | – | – | – |
| + Diffusion | 70.86 | 12.23 | 132.65 | 10.74 | 368.99 | -2.96 | – | – | – | – | – | – |
| + GSA | **69.76** | **10.48** | **129.67** | **8.49** | **362.11** | **-4.77** | **596.43** | – | **907.65** | – | **1125.04** | – |

*"–" indicates that the model was unable to produce a result for a given instance either within the 90-minute time limit or due to GPU memory constraints.*

### 5.3 Inference on Real-World Dataset

To assess the generalization ability of our model, we evaluate it on real-world benchmark datasets, TSPLib [1] and CVRPLib [47].

Following prior works [54, 34, 35, 56], we evaluate our model on TSPLib with test instances ranging from 50 to 200 nodes and from 200 to 1000 nodes, respectively. We adopt the same experimental setup and baseline methods as described in Section 5.1. The results are reported in Table 3 in the Appendix. Our model outperforms GFACS and AGFN in gap by $51.45\%$ and $25.97\%$ on 200-500 node instances, and $65.78\%$ and $30.69\%$ on 500-1000 node instances, respectively. Compared to the diffusion-based models with two-opt post-processing, DEG achieves superior gap performance over DIFUSCO, DEITSP, T2T, and Fast T2T by $88.59\%$, $84.56\%$, $72.73\%$ and $74.07\%$ on 200-500 node instances, and $70.72\%$, $45.11\%$, $19.84\%$ and $30.82\%$ on 500-1000 node instances. Overall, our model outperforms both GFlowNet-based models and diffusion-based solvers on TSPLib, across both 50-200 and 200-1000 instances.

We test CVRPLib containing instances with the number of node ranging from 3000 to 16000. The experiment setting and baselines are same with comparison on CVRP benchmarks. As shown in Table 4, DEG achieves consistently lower optimality gaps than all baseline methods across a wide range of real-world CVRPLib instances. Compared to GFACS, DEG reduces the gap on the same set of instances. Compared to AGFN, our model reduces significant gap on the same set of instances. Relative to LEHD, DEG achieves gap improvements of $0.84\%$ on L2, $20.89\%$ on A1, $9.85\%$ on A2, $14.87\%$ on G1, $36.51\%$ on G2, $52.59\%$ on B1, and $42.42\%$ on B2, while LEHD performs slightly better on L1. Finally, compared to POMO, our model significantly reduces the gap by $73.92\%$, $66.63\%$, $86.68\%$, and $85.05\%$ on L1, L2, A1, and A2. These results demonstrate the superior scalability and generalization ability of DEG in solving large-scale real-world CVRP instances.

### 5.4 Ablation Study

**Component Evaluation.** We evaluate the contribution of each key component in DEG through an ablation study. We begin by training the backbone model, which involves learning a pure GFlowNet to generate trajectories, serving as the baseline for subsequent comparison. Next, we integrate the diffusion mechanism into the backbone by incorporating the reward-level feedback and noise schedule

described in Sec.4.1, in order to evaluate the effectiveness of the diffusion component. Finally, the GSA decoder (Sec. 4.3) is applied on top of the previous model to assess its impact.

The results are presented in Table 5. Incorporating diffusion into the backbone leads to significant performance improvements. Adding the GSA module on top of the diffusion-enhanced model further improves performance on the TSP and CVRP across the same instance sizes. In addition to improving performance, GSA also reduces computational resource consumption by shortening the observation distance and decreasing the number of connections per node. As a result, the model is able to efficiently solve 10000-node TSP instances as well as 5000-, 8000- and 10000-node CVRP instances.

## 6 Conclusion

In this paper, we introduce Diffusion-Enhanced GFlowNet (DEG), a generative framework that integrates GFlowNet with diffusion to overcome key limitations of existing GFlowNet-based solvers for vehicle routing problems (VRPs). Specifically, their insufficient flow expansion in high-reward regions. To enable this integration, DEG leverages the intrinsic diversity of GFlowNet to provide edge-specific backward signals as a noise basis for diffusion integration. It then employs the diffusion's noise schedule to inject noise into these signals, and denoises them through the GFlowNet paradigm, guiding sufficient flow expansion toward high-reward regions to derive better solutions. Furthermore, to enhance scalability and generalization ability, we incorporate a Graph-Scale Adapter (GSA) that enables effective adaptation across diverse instance sizes. Extensive experiments on synthetic benchmarks and real-world datasets, including instances with up to 10,000 nodes, show that DEG delivers favorable performance compared to diffusion-, GFlowNet-based and other neural baselines. These results demonstrate that DEG has strong scalability, efficient inference, and robust generalization for solving VRPs.

While DEG outperforms the strong AR model like LEHD in inference time and generalization, one of the **limitations** is that DEG's parallel decoding still falls short in small-scale settings, where AR models excel at fine-grained path construction. Bridging this gap remains an important direction for our future work, which involves enhancing the local decision-making capacity of NAR models. In addition, extending DEG to a wider range of VRP variants is another venue we plan to explore.

## Acknowledgment

This research is supported by the Ministry of Education, Singapore under its Academic Research Fund (AcRF) Tier 2 Grant Call (Proposal ID: T2EP20225-0056), and its Academic Research Funding Tier 3 (Proposal ID: MOET32025-0010). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of the Ministry of Education, Singapore. This research is also supported by the Lee Kong Chian Fellowship awarded to CAO Zhiguang by Singapore Management University.

## References


[1] Florian Arnold, Michel Gendreau, and Kenneth Sörensen. Efficiently solving very large-scale routing problems. *Computers & operations research*, 107:32–42, 2019.

[2] Barrie M Baker and MA1951066 Ayechew. A genetic algorithm for the vehicle routing problem. *Computers & Operations Research*, 30(5):787–800, 2003.

[3] John E Bell and Patrick R McMullen. Ant colony optimization techniques for the vehicle routing problem. *Advanced engineering informatics*, 18(1):41–48, 2004.

[4] Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394, 2021.

[5] Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.

[6] DD Bochtis and Claus G Sørensen. The vehicle routing problem in field logistics part i. *Biosystems engineering*, 104(4):447–457, 2009.

[7] DD Bochtis and Claus G Sørensen. The vehicle routing problem in field logistics: Part ii. *Biosystems engineering*, 105(2):180–188, 2010.

[8] Olli Bräysy and Michel Gendreau. Genetic algorithms for the vehicle routing problem with time windows. *Arpakannus,(1)*, pages 33–38, 2001.

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

[10] Francesco Carrabs, Raffaele Cerulli, and Anna Sciomachen. An exact approach for the grocery delivery problem in urban areas. *Soft Computing*, 21:2439–2450, 2017.

[11] Diego Cattaruzza, Nabil Absi, Dominique Feillet, and Jesús González-Feliu. Vehicle routing problems for city logistics. *EURO Journal on Transportation and Logistics*, 6(1):51–79, 2017.

[12] Yihang Chen and Lukas Mauch. Order-preserving gflownets. In *The Twelfth International Conference on Learning Representations*.

[13] Wen-Chyuan Chiang and Robert A Russell. Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Annals of Operations Research*, 63:3–27, 1996.

[14] Goran Ćirović, Dragan Pamučar, and Darko Božanić. Green logistic vehicle routing problem: Routing light delivery vehicles in urban areas using a neuro-fuzzy model. *Expert Systems with Applications*, 41(9):4245–4258, 2014.

[15] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[16] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11, 2018.

[17] Sotiris P Gayialis, Grigorios D Konstantakopoulos, and Ilias P Tatsiopoulos. Vehicle routing problem for urban freight transportation: A review of the recent literature. In *Operational Research in the Digital Era–ICT Challenges: 6th International Symposium and 28th National Conference on Operational Research, Thessaloniki, Greece, June 2017*, pages 89–104. Springer, 2019.

[18] Alexandros Graikos, Nikolay Malkin, Nebojsa Jojic, and Dimitris Samaras. Diffusion models as plug-and-play priors. *Advances in Neural Information Processing Systems*, 35:14715–14728, 2022.

[19] Jun Han and Claudio Moraga. The influence of the sigmoid function parameters on the speed of backpropagation learning. In *International workshop on artificial neural networks*, pages 195–201. Springer, 1995.

[20] Keld Helsgaun. An effective implementation of the lin–kernighan traveling salesman heuristic. *European journal of operational research*, 126(1):106–130, 2000.

[21] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[22] Yu-Hsiang Hsiao, Mu-Chen Chen, Kuan-Yu Lu, and Cheng-Lin Chin. Last-mile distribution planning for fruit-and-vegetable cold chains. *The International Journal of Logistics Management*, 29(3):862–886, 2018.

[23] Rongjie Huang, Jiawei Huang, Dongchao Yang, Yi Ren, Luping Liu, Mingze Li, Zhenhui Ye, Jinglin Liu, Xiang Yin, and Zhou Zhao. Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models. In *International Conference on Machine Learning*, pages 13916–13932. PMLR, 2023.

[24] Ziwei Huang, Jianan Zhou, Zhiguang Cao, and Yixin Xu. Rethinking light decoder-based solvers for vehicle routing problems. In *International Conference on Learning Representations (ICLR)*, 2025.

[25] Zarif Ikram, Ling Pan, and Dianbo Liu. Evolution guided generative flow networks. In *ICLR 2024 Workshop on Generative and Experimental Perspectives for Biomolecular Design*.

[26] Moksh Jain, Emmanuel Bengio, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Bonaventure FP Dossou, Chanakya Ajit Ekbote, Jie Fu, Tianyu Zhang, Michael Kilgour, Dinghuai Zhang, et al. Biological sequence design with gflownets. In *International Conference on Machine Learning*, pages 9786–9801. PMLR, 2022.

[27] Minsu Kim, Sanghyeok Choi, Hyeonah Kim, Jiwoo Son, Jinkyoo Park, and Yoshua Bengio. Ant colony sampling with gflownets for combinatorial optimization. In *The 28th International Conference on Artificial Intelligence and Statistics*, 2025.

[28] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*.

[29] Grigorios D Konstantakopoulos, Sotiris P Gayialis, and Evripidis P Kechagias. Vehicle routing problem and related algorithms for logistics distribution: A literature review and classification. *Operational research*, 22(3):2033–2062, 2022.

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

[31] Yiyo Kuo. Using simulated annealing to minimize fuel consumption for the time-dependent vehicle routing problem. *Computers & Industrial Engineering*, 59(1):157–165, 2010.

[32] Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning. *Advances in Neural Information Processing Systems*, 33:21188–21198, 2020.

[33] Kaijun Leng and Shanghong Li. Distribution path optimization for intelligent logistics vehicles of urban rail transportation using vrp optimization model. *IEEE Transactions on Intelligent Transportation Systems*, 23(2):1661–1669, 2021.

[34] Yang Li, Jinpei Guo, Runzhong Wang, and Junchi Yan. T2t: From distribution learning in training to gradient search in testing for combinatorial optimization. *Advances in Neural Information Processing Systems*, 36:50020–50040, 2023.

[35] Yang Li, Jinpei Guo, Runzhong Wang, Hongyuan Zha, and Junchi Yan. Fast t2t: Optimization consistency speeds up diffusion-based training-to-testing solving for combinatorial optimization. *Advances in Neural Information Processing Systems*, 37:30179–30206, 2024.

[36] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

[37] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2022.

[38] Fu Luo, Xi Lin, Fei Liu, Qingfu Zhang, and Zhenkun Wang. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. *Advances in Neural Information Processing Systems*, 36:8845–8864, 2023.

[39] Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35:5955–5967, 2022.

[40] Mazin Abed Mohammed, Mohd Khanapi Abd Ghani, Raed Ibraheem Hamed, Salama A Mostafa, Mohd Sharifuddin Ahmad, and Dheyaa Ahmed Ibrahim. Solving vehicle routing problem by using improved genetic algorithm for optimal solution. *Journal of computational science*, 21:255–262, 2017.

[41] Roberto Montemanni, Luca Maria Gambardella, Andrea Emilio Rizzoli, and Alberto V Donati. Ant colony system for a dynamic vehicle routing problem. *Journal of combinatorial optimization*, 10:327–343, 2005.

[42] Douglas C Montgomery and George C Runger. *Applied statistics and probability for engineers*. John wiley & sons, 2010.

[43] Andrei Cristian Nica, Moksh Jain, Emmanuel Bengio, Cheng-Hao Liu, Maksym Korablyov, Michael M Bronstein, and Yoshua Bengio. Evaluating generalization in gflownets for molecule design. In *ICLR2022 Machine Learning for Drug Discovery*, 2022.

[44] Ibrahim Hassan Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of operations research*, 41:421–451, 1993.

[45] Mingjun Pan, Guanquan Lin, You-Wei Luo, Bin Zhu, Zhien Dai, Lijun Sun, and Chun Yuan. Preference optimization for combinatorial optimization problems. In *Forty-second International Conference on Machine Learning*, 2025.

[46] Christian Prins. Two memetic algorithms for heterogeneous fleet vehicle routing problems. *Engineering Applications of Artificial Intelligence*, 22(6):916–928, 2009.

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

[48] Andrea Emilio Rizzoli, Roberto Montemanni, Enzo Lucibello, and Luca Maria Gambardella. Ant colony optimization for real-world vehicle routing problems: from theory to applications. *Swarm Intelligence*, 1:135–151, 2007.

[49] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[50] Sheldon M Ross. *Introduction to probability models*. Academic press, 2014.

[51] Arne Schneuing, Charles Harris, Yuanqi Du, Kieran Didi, Arian Jamasb, Ilia Igashov, Weitao Du, Carla Gomes, Tom L Blundell, Pietro Lio, et al. Structure-based drug design with equivariant diffusion models. *Nature Computational Science*, 4(12):899–909, 2024.

[52] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015.

[53] Hartej Soin, Tapas Tripura, and Souvik Chakraborty. Generative flow induced neural architecture search: Towards discovering optimal architecture in wavelet neural operator. *CoRR*, 2024.

[54] Zhiqing Sun and Yiming Yang. Difusco: Graph-based diffusion solvers for combinatorial optimization. *Advances in neural information processing systems*, 36:3706–3731, 2023.

[55] Amalia Utamima and Arif Djunaidy. Agricultural routing planning: A narrative review of literature. *Procedia Computer Science*, 197:693–700, 2022.

[56] Mingzhao Wang, You Zhou, Zhiguang Cao, Yubin Xiao, Xuan Wu, Wei Pang, Yuan Jiang, Hui Yang, Peng Zhao, and Yuanshu Li. 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*, 2025.

[57] Daqing Wu, Jiyu Li, Jiye Cui, and Dong Hu. Research on the time-dependent vehicle routing problem for fresh agricultural products based on customer value. *Agriculture*, 13(3):681, 2023.

[58] Dan Xie, Zhi-Wei Wang, Zhen-Yi Gu, Wan-Yue Diao, Fang-Yu Tao, Chang Liu, Hai-Zhu Sun, Xing-Long Wu, Jia-Wei Wang, and Jing-Ping Zhang. Polymeric molecular design towards horizontal zn electrodeposits at constrained 2d zn2+ diffusion: dendrite-free zn anode for long-life and high-rate aqueous zinc metal battery. *Advanced Functional Materials*, 32(32):2204066, 2022.

[59] Liang Xin, Wen Song, Zhiguang Cao, and Jie Zhang. 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, 2021.

[60] Qi Yao, Shenjun Zhu, and Yanhui Li. Green vehicle-routing problem of fresh agricultural products considering carbon emission. *International Journal of Environmental Research and Public Health*, 19(14):8675, 2022.

[61] Haoran Ye, Jiarui Wang, Zhiguang Cao, Helan Liang, and Yong Li. Deepaco: Neural-enhanced ant systems for combinatorial optimization. *Advances in neural information processing systems*, 36:43706–43728, 2023.

[62] Hang Yi, Ziwei Huang, Yining Ma, and Zhiguang Cao. Radar: Learning to route with asymmetry-aware distance representations. In *International Conference on Learning Representations (ICLR)*, 2026.

[63] David W Zhang, Corrado Rainone, Markus Peschl, and Roberto Bondesan. Robust scheduling with gflownets. In *The Eleventh International Conference on Learning Representations*.

[64] Dinghuai Zhang, Ricky TQ Chen, Cheng-Hao Liu, Aaron Courville, and Yoshua Bengio. Diffusion generative flow samplers: Improving learning signals through partial trajectory optimization. In *The Twelfth International Conference on Learning Representations*.

[65] Dinghuai Zhang, Hanjun Dai, Nikolay Malkin, Aaron C Courville, Yoshua Bengio, and Ling Pan. Let the flows tell: Solving graph combinatorial problems with gflownets. *Advances in neural information processing systems*, 36:11952–11969, 2023.

[66] Ni Zhang and Zhiguang Cao. Hybrid-balance gflownet for solving vehicle routing problems. *Advances in Neural Information Processing Systems*, 2025.

[67] Ni Zhang, Zhiguang Cao, Jianan Zhou, Cong Zhang, and Yew-Soon Ong. An agentic framework with LLMs for solving complex vehicle routing problems. In *International Conference on Learning Representations (ICLR)*, 2026.

[68] Ni Zhang, Jingfeng Yang, Zhiguang Cao, and Xu Chi. Adversarial generative flow network for solving vehicle routing problems. In *The Thirteenth International Conference on Learning Representations*, 2025.

[69] Yiheng Zhu, Jialu Wu, Chaowen Hu, Jiahuan Yan, Tingjun Hou, Jian Wu, et al. Sample-efficient multi-objective molecular optimization with gflownets. *Advances in Neural Information Processing Systems*, 36:79667–79684, 2023.

# A More Methodology Detail

## A.1 Forward Network and Training Paradigm

Following prior work [27, 68, 54, 59], we adopt a Graph Neural Network (GNN) as the forward network $\theta$, incorporating distance information as edge features. GNN employs a sparsification strategy on the input graph $\mathcal{G}$ to reduce computational complexity. Specifically, the graph retains only the $w$ shortest edges per node, resulting in a sparse graph $\mathcal{G}^* = (\mathcal{V}, \mathcal{E}^*)$. The node coordinates $\mathbf{u}_v \in \mathbb{R}^2$ and edge distances $\mathbf{u}_e \in \mathbb{R}$ are first embedded into initial node and edge features $\mathbf{h}_i^0, \mathbf{e}_{ij}^0 \in \mathbb{R}^d$ by learnable linear layers, where $d$ represents the number of hidden units per layer. Then, $\mathbf{h}_i^0$ and $\mathbf{e}_{ij}^0$ are processed by GNN to capture the underlying relationships between nodes and edges. At each layer $l$, node and edge features are updated as:

$$\mathbf{h}_i^{l+1} = \mathbf{h}_i^l + \mathrm{ACT}(\mathrm{BN}(\mathbf{U}^l \mathbf{h}_i^l + \mathcal{A}_{j\in\mathcal{N}_i}(\sigma(\mathbf{e}_{ij}^l) \odot \mathbf{V}^l \mathbf{h}_j^l))), \tag{13}$$

$$\mathbf{e}_{ij}^{l+1} = \mathbf{e}_{ij}^l + \mathrm{ACT}(\mathrm{BN}(\mathbf{P}^l \mathbf{e}_{ij}^l + \mathbf{Q}^l \mathbf{h}_i^l + \mathbf{R}^l \mathbf{h}_j^l)). \tag{14}$$

Here, $\mathbf{U}^l, \mathbf{V}^l, \mathbf{P}^l, \mathbf{Q}^l, \mathbf{R}^l \in \mathbb{R}^{d\times d}$ are learnable parameters of the GNN layers; SiLU [16] is used as the activation function for all intermediate layers; BN denotes batch normalization; $\sigma$ is the sigmoid function [19]; $\odot$ is the Hadamard product; and $\mathcal{A}$ denotes mean pooling over neighbors.

The denoised forward policy $\eta(\mathcal{G}^*, \boldsymbol{\theta})$ is computed using a multi-layer perceptron (MLP) applied to the final-layer GNN embeddings, followed by a sigmoid activation to normalize outputs. The source flow $Z(\mathcal{G}; \theta)$ is computed from the final-layer edge embedding $e^d$ as follows:

$$Z(\mathcal{G}; \theta) = M \cdot \mathrm{ReLU}(J \cdot e^d + b) + f, \tag{15}$$

where $M$, $J$, $b$, and $f$ are learnable parameters, and ReLU is the activation function.

# B Analysis of the Learning Target under the Noise Schedule

We describe the implemented signal construction, then derive the population target at each noise level, analyze the two endpoints, and bound its distance from reward-proportional sampling.

## B.1 Setup and well-posedness

Fix an instance graph $\mathcal{G}$. A trajectory $\tau$ is a complete construction sequence with reward $R(\tau) > 0$, and $|\tau|$ denotes its number of edges. For TSP, $|\tau| = n$ for every tour. For CVRP, $|\tau| = n + K(\tau)$, where $K(\tau)$ is the number of routes.

**Edge signal.** The edge reward $\bar{R}(e)$ is the average reward of sampled trajectories containing edge $e$. After z-score normalization, the clean signal is $x_0[e] = \Phi(\hat{R}(e)) \in (0, 1)$ (Eq. 5).

**Corruption.** At noise level $t$ (Eq. 6),

$$x_t[e] = \sqrt{\bar{\alpha}_t}\, x_0[e] + \sqrt{1-\bar{\alpha}_t}\, \epsilon[e], \qquad \epsilon[e] \sim \mathcal{N}(0,1), \qquad \bar{\alpha}_t = \prod_{k=1}^{t}(1-\beta_k), \tag{16}$$

where the noise is sampled independently across edges.

**Clamping.** Because $x_t[e]$ may fall outside $[0, 1]$, we use

$$\tilde{x}_t[e] = \max\big(\mathrm{clamp}(x_t[e], 0, 1), \delta\big) \in [\delta, 1], \qquad \delta = 10^{-10}. \tag{17}$$

**Backward-side trajectory weight.** The backward term used in TB is

$$W_t(\tau) = \prod_{e\in\tau} \tilde{x}_t[e] \in [\delta^{|\tau|}, 1]. \tag{18}$$

The factors $\tilde{x}_t[e]$ are used directly rather than normalized over parent states. Thus, $W_t$ is a positive trajectory weight rather than a normalized backward transition kernel. The clamping guarantees $W_t > 0$.

**Objective.** The TB loss for a trajectory is

$$L_t(\tau;\theta) = \Big(\log Z(\mathcal{G};\theta) + \log P_F(\tau;\theta) - \log R(\tau) - \log W_t(\tau)\Big)^2. \tag{19}$$

Since $W_t > 0$ and $\log W_t$ is bounded, the loss is well defined. Equivalently, this is TB with the trajectory reward $\tilde{R}_t(\tau) = R(\tau)W_t(\tau)$.

**Assumption 1** (Population setting)**.** *The signal $x_0$ is fixed, trajectories are sampled from a distribution with full support, and optimization is over all strictly positive trajectory distributions $P_F$ together with the normalizer $Z > 0$.*

### B.2 Exact target at every noise level

**Lemma 1** (Bias–variance decomposition)**.** *For every trajectory $\tau$ and noise level $t$,*

$$\mathbb{E}_\epsilon\big[L_t(\tau;\theta)\big] = \Big(\log Z(\mathcal{G};\theta) + \log P_F(\tau;\theta) - \log R(\tau) - \mu_t(\tau)\Big)^2 + \sigma_t^2(\tau), \tag{20}$$

*where*

$$\mu_t(\tau) = \mathbb{E}_\epsilon\big[\log W_t(\tau)\big] = \sum_{e\in\tau} h_t\big(x_0[e]\big), \tag{21}$$

*with*

$$h_t(u) = \mathbb{E}_{\epsilon\sim\mathcal{N}(0,1)}\left[\log\max\left(\text{clamp}\left(\sqrt{\bar{\alpha}_t}\,u + \sqrt{1-\bar{\alpha}_t}\,\epsilon, 0, 1\right), \delta\right)\right], \tag{22}$$

*and $\sigma_t^2(\tau) = \text{Var}_\epsilon[\log W_t(\tau)]$, which does not depend on $\theta$.*

*Proof.* Using $\mathbb{E}[(a-X)^2] = (a - \mathbb{E}[X])^2 + \text{Var}(X)$, set

$$a = \log Z + \log P_F(\tau), \qquad X = \log R(\tau) + \log W_t(\tau).$$

For fixed $\tau$, $R(\tau)$ is deterministic, while $\log W_t(\tau) \in [|\tau|\log\delta, 0]$. The result follows from $\log W_t(\tau) = \sum_{e\in\tau}\log\tilde{x}_t[e]$ and linearity of expectation. □

**Proposition 1** (Exact target)**.** *Under Assumption 1, the expected TB loss at noise level $t$ has the unique population minimizer*

$$P_F^t(\tau) = \frac{R(\tau)e^{\mu_t(\tau)}}{Z_t}, \qquad Z_t = \sum_{\tau'} R(\tau')e^{\mu_t(\tau')}. \tag{23}$$

*Proof.* By Lemma 1, the $\theta$-dependent term is minimized when

$$ZP_F(\tau) = R(\tau)e^{\mu_t(\tau)}$$

for every trajectory. Full support requires this equality for all $\tau$. Summing over trajectories gives $Z = Z_t$, yielding Eq. 23. □

**Remark 1.** *Proposition 1 is exact. The injected noise affects the optimum through $\mathbb{E}_\epsilon[\log W_t]$, while its variance contributes only the $\theta$-independent term $\sigma_t^2$. The result is independent of the trajectory sampling distribution as long as it has full support, including on-policy sampling. When $x_0$ changes during training, the proposition describes the population target associated with the current $x_0$ and noise level. It does not imply convergence of the parameterized training dynamics.*

### B.3 Low-noise endpoint: exploitation

At $\bar{\alpha}_t = 1$,

$$\tilde{x}_0[e] = \max(x_0[e], \delta), \qquad h_0(u) = \log\max(u, \delta).$$

Define

$$W_0(\tau) = \prod_{e\in\tau}\max(x_0[e], \delta).$$

When $x_0[e] \geq \delta$ along $\tau$, this reduces to $\prod_{e\in\tau} x_0[e]$. Proposition 1 then gives

$$P_F^0(\tau) \propto R(\tau)W_0(\tau). \tag{24}$$

**Assumption 2** (Edge decomposability)**.** *There exists a constant $c > 0$ such that*

$$\prod_{e \in \tau} x_0[e] = cR(\tau)$$

*for every trajectory $\tau$.*

**Proposition 2** (Low-noise endpoint)**.** *(a) Under Assumption 2 and $x_0[e] \geq \delta$,*

$$P_F^0(\tau) \propto R(\tau)^2.$$

*(b) Without Assumption 2, let $\pi_R(\tau) \propto R(\tau)$. Then*

$$\mathbb{E}_{P_F^0}[R] - \mathbb{E}_{\pi_R}[R] = \frac{\mathrm{Cov}_{\pi_R}(R, W_0)}{\mathbb{E}_{\pi_R}[W_0]}. \tag{25}$$

*Proof.* For (a), substitute $W_0(\tau) = cR(\tau)$ into Eq. 24. For (b),

$$P_F^0(\tau) = \frac{\pi_R(\tau) W_0(\tau)}{\mathbb{E}_{\pi_R}[W_0]}.$$

Therefore,

$$\mathbb{E}_{P_F^0}[R] = \frac{\mathbb{E}_{\pi_R}[RW_0]}{\mathbb{E}_{\pi_R}[W_0]},$$

which gives Eq. 25. □

Our CDF-based construction of $x_0$ does not satisfy Assumption 2 exactly, so the $R^2$ form is a special case rather than a general property. Equation 25 requires no such assumption: the low-noise target has higher expected reward than reward-proportional sampling whenever $\mathrm{Cov}_{\pi_R}(R, W_0) > 0$. This is precisely the behavior encouraged by constructing $x_0$ from edge-reward statistics.

### B.4 High-noise endpoint: exploration

**Proposition 3** (High-noise endpoint)**.** *At $\bar{\alpha}_t = 0$,*

$$\mu_T(\tau) = c_T |\tau|,$$

*where*

$$c_T = \mathbb{E}_\epsilon \left[ \log \max(\mathrm{clamp}(\epsilon, 0, 1), \delta) \right].$$

*Hence,*

$$P_F^T(\tau) \propto R(\tau)$$

*for TSP, while for CVRP,*

$$P_F^T(\tau) \propto R(\tau) e^{c_T K(\tau)}.$$

*Proof.* At $\bar{\alpha}_t = 0$, $\tilde{x}_T[e]$ depends only on $\epsilon[e]$ and is independent of $x_0[e]$. Thus $h_T(u) = c_T$ for all $u$, giving $\mu_T(\tau) = c_T |\tau|$.

For TSP, every tour contains $n$ edges, so $e^{c_T n}$ is absorbed into the normalizer. For CVRP, $|\tau| = n + K(\tau)$; after absorbing $e^{c_T n}$, the remaining factor is $e^{c_T K(\tau)}$. □

**Remark 2** (CVRP)**.** *For the implemented value $\delta = 10^{-10}$, numerical integration gives $c_T \approx -11.9$. The CVRP endpoint is therefore reward-proportional among solutions with the same number of routes, while favoring fewer routes across groups with different $K(\tau)$.*

*On the evaluated samples ($n = 500$ and $n = 1000$, 32 instances each), route count and cost are positively correlated on every instance, with mean correlations of $+0.53$ and $+0.36$, respectively. This does not imply that fewer routes are always preferable or that near-optimal solutions always use the minimum number of routes. Therefore, the unconditional reward-proportional endpoint is exact for TSP only; for CVRP, the exploration benefit is evaluated empirically.*

### B.5 Rate between the endpoints

**Lemma 2** (Monotonicity and Lipschitz constant of $h_t$)**.** *Let $\bar{\alpha}_t < 1$. For $0 \le u \le v \le 1$,*

$$0 \le h_t(v) - h_t(u) \le K_t(v-u), \qquad K_t \le \frac{\sqrt{\bar{\alpha}_t}\log(1/\delta)}{\sqrt{2\pi(1-\bar{\alpha}_t)}}. \tag{26}$$

*Proof.* Let $s_t = \sqrt{1-\bar{\alpha}_t}$ and

$$\ell(z) = \log\max(\mathrm{clamp}(z, 0, 1), \delta).$$

Then

$$h_t(u) = \mathbb{E}_\epsilon\left[\ell(\sqrt{\bar{\alpha}_t}u + s_t\epsilon)\right].$$

Since $\ell$ is nondecreasing, $h_t(v) \ge h_t(u)$. Moreover, $\ell'(z) = 1/z$ on $(\delta, 1)$ and $0$ elsewhere. Therefore,

$$h_t'(u) = \sqrt{\bar{\alpha}_t}\int_\delta^1 \frac{1}{z}\frac{1}{s_t}\varphi\left(\frac{z-\sqrt{\bar{\alpha}_t}u}{s_t}\right)dz \le \frac{\sqrt{\bar{\alpha}_t}\log(1/\delta)}{s_t\sqrt{2\pi}}, \tag{27}$$

where $\varphi$ is the standard normal density and $\varphi \le 1/\sqrt{2\pi}$. □

For $\delta = 10^{-10}$, $\log(1/\delta) \approx 23.0$, so the constant is explicit.

**Theorem 1** (Distance to reward-proportional sampling, TSP)**.** *For TSP with $n$ nodes and $\bar{\alpha}_t < 1$, let $P_F^t$ be the target in Proposition 1 and $\pi_R(\tau) \propto R(\tau)$. Then*

$$\mathrm{KL}\big(P_F^t \,\|\, \pi_R\big) \le nK_t \le \frac{n\sqrt{\bar{\alpha}_t}\log(1/\delta)}{\sqrt{2\pi(1-\bar{\alpha}_t)}} \xrightarrow[\bar{\alpha}_t\to 0]{} 0. \tag{28}$$

*Proof.* Let

$$g(\tau) = \log\frac{P_F^t(\tau)}{\pi_R(\tau)} = \mu_t(\tau) - \log\frac{Z_t}{Z_R}, \qquad Z_R = \sum_\tau R(\tau).$$

Each TSP tour contains $n$ edges. By Lemma 2, each term $h_t(x_0[e])$ varies over an interval of length at most $K_t$. Hence, $\mu_t(\tau)$ varies by at most $nK_t$, and so does $g(\tau)$.

Because both distributions are normalized, $\mathbb{E}_{\pi_R}[e^g] = 1$, which implies $\inf_\tau g \le 0 \le \sup_\tau g$. Therefore,

$$\mathrm{KL}(P_F^t \,\|\, \pi_R) = \mathbb{E}_{P_F^t}[g] \le \sup_\tau g \le nK_t.$$

□

**Signal correlation.** Let $\sigma_0^2$ denote the variance of $x_0[e]$ over edges. Since $\epsilon$ is independent of $x_0$,

$$\mathrm{Cov}(x_t, x_0) = \sqrt{\bar{\alpha}_t}\sigma_0^2,$$

and

$$\mathrm{Var}(x_t) = \bar{\alpha}_t\sigma_0^2 + 1 - \bar{\alpha}_t.$$

Thus,

$$\rho(t) = \sqrt{\frac{\bar{\alpha}_t\sigma_0^2}{\bar{\alpha}_t\sigma_0^2 + 1 - \bar{\alpha}_t}}, \tag{29}$$

which decreases monotonically from 1 to 0.

As $\bar{\alpha}_t \to 0$, $\rho(t) = \Theta(\sqrt{\bar{\alpha}_t})$, so Eq. 28 vanishes at rate $O(n\rho(t))$. The dependence on $n$ also shows that the bound is mainly informative late in the schedule, when $\sqrt{\bar{\alpha}_t} \ll 1/n$. Since $\bar{\alpha}_t$ decreases with $t$, the bound tightens throughout training. The edge-dependent reward signal is gradually weakened, and the TSP target converges to reward-proportional sampling at the endpoint. For CVRP, the same argument applies among trajectories with the same number of routes.

### B.6 Trajectories and solutions

$P_F$ is defined over trajectories rather than solutions. For TSP, each tour has the same trajectory multiplicity—between 1 and $2n$, depending on the convention for starting node and direction. This common factor cancels, so reward proportionality over trajectories also holds over solutions.

For CVRP, the number of trajectories representing a solution can depend on its route structure. This introduces the same qualification as the factor $e^{c_T K(\tau)}$ in Remark 2; the issue disappears when comparing solutions with the same route structure.

## C More Experiment Detail

### C.1 GSA Comparison Results

We validate the behavior that the best outcomes of larger-scale instances often appear at earlier training stages. We evaluate the model on datasets of varying sizes (200, 500, and 1000 nodes), each containing 128 test cases. The model is trained on 100-node instances and evaluated using checkpoints saved at different training epochs, with each epoch comprising 20 training steps. The results are presented in Fig. 2. As the instance size increases from 200 to 500 and then to 1000 nodes, the performance peak occurs progressively earlier during training. Moreover, the model's performance degrades significantly after reaching its peak, especially on larger-scale instances.

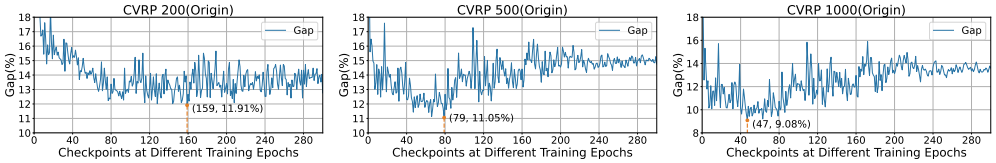



Figure 2: The original performance of DEG test on different scale instance. Gap (%) is computed relative to LKH, and lower values indicate better results.

After using GSA, as illustrated in Fig. 3, GSA successfully constrains the optimal inference checkpoints across different instance sizes into a consistent interval. Moreover, as training progresses, the model shows a steadily improving performance at later checkpoints—unlike the original decoder, which tends to degrade after early peaks. In addition, GSA achieves overall better performance compared to the baseline setup without GSA. These results indicate that GSA mitigates spatial pattern overfitting to the training instance size and enables the model to derive consistent results across varying instance scales.

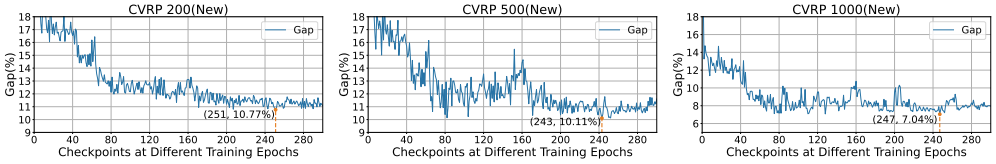



Figure 3: The performance of GSA-based DEG test on different scale instance. Gap (%) is computed relative to LKH, and lower values indicate better results.

### C.2 TSP comparison on Diffusion-Based Solver with Two-Opt

We compare DEG with existing diffusion-based models, all enhanced with Two-Opt post-processing, and the results are summarized in Table 6. Our method, DEG, achieves an objective value reduction of $2.42\%$, $2.27\%$, and $2.74\%$ on 500-, 1000-, 3000-node instance over DIFUSCO. Compared to Fast-T2T, DEG yields improvements of $0.88\%$, $1.66\%$, and $4.17\%$ on the same instance sizes. Furthermore, DEG outperforms DEITSP by $0.80\%$, $2.04\%$ on 500-, 1000-, 3000-node instances. Notably, DEG is capable of scaling to larger problem sizes, specifically 5,000, 8,000 and 10,000 nodes, whereas other diffusion-based models fail to run due to memory limitations.

Table 6: Comparison of performance on TSP with varying node sizes with two-opt post processing. Obj. denotes the total route length and Gap (%) is computed relative to LKH. For all metrics, lower values indicate better results.

| TSP | 500 | | 1000 | | 3000 | | 5000 | | 8000 | | 10000 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Obj. | Gap(%) | Obj. | Gap(%) | Obj. | Gap(%) | Obj. | Gap(%) | Obj. | Gap(%) | Obj. | Gap(%) |
| DIFUSCO (NeurIPS'23) | 17.32 | 4.65 | 24.22 | 4.66 | 43.41 | 7.69 | – | – | – | – | – | – |
| T2T (NeurIPS'23) | 17.14 | 3.56 | 24.18 | 4.49 | 42.97 | 6.60 | – | – | – | – | – | – |
| Fast T2T (NeurIPS'24) | 17.05 | 3.02 | 24.07 | 4.02 | 44.06 | 9.30 | – | – | – | – | – | – |
| DEITSP (KDD'25) | 16.98 | 2.60 | 23.86 | 3.15 | 43.10 | 6.92 | – | – | – | – | – | – |
| DEG | **16.90** | **2.11** | **23.67** | **2.30** | **42.22** | **4.74** | **54.61** | **5.34** | **69.46** | **5.42** | **80.05** | **5.69** |

*"–" indicates that the model was unable to produce a result for a given instance either within the 90-minute time limit or due to GPU memory constraints.*

Table 7: Comparison on small-scale TSP and CVRP instances (100 and 200 Nodes). Obj. denotes the total route length; Gap (%) is computed relative to LKH; and Time (s) indicates the average time required to solve a single instance. For all metrics, lower values indicate better results.

| TSP | 100 | | | 200 | | |
|---|---|---|---|---|---|---|
| | Obj. | Time(s) | Gap(%) | Obj. | Time(s) | Gap(%) |
| LKH | 7.76 | 1.50 | - | 10.62 | 9.80 | - |
| GFACS | 8.80 | 0.53 | 13.54 | 13.20 | 1.15 | 24.29 |
| AGFN | 8.49 | 0.05 | 9.55 | 11.82 | 0.08 | 11.30 |
| DEG | **8.40** | 0.06 | **8.25** | **11.65** | 0.10 | **9.42** |
| DIFUSCO (w/ LS) | 7.88 | 0.42 | 1.68 | 10.80 | 1.53 | 1.70 |
| T2T (w/ LS) | 7.80 | 0.37 | 0.65 | 10.80 | 1.15 | 1.72 |
| DEITSP (w/ LS) | **7.78** | 0.08 | **0.39** | 10.76 | 1.20 | 1.32 |
| Fast T2T (w/ LS) | 7.82 | 0.11 | 0.90 | 10.78 | 1.19 | 1.51 |
| DEG (w/ LS) | 7.79 | 0.07 | 0.52 | **10.76** | 1.15 | **1.32** |

| CVRP | 100 | | | 200 | | |
|---|---|---|---|---|---|---|
| | Obj. | Time(s) | Gap(%) | Obj. | Time(s) | Gap(%) |
| LKH | 15.57 | 18.80 | - | 28.04 | 50.83 | - |
| GFACS | 19.26 | 1.90 | 23.70 | 35.36 | 3.14 | 26.11 |
| AGFN | 17.78 | 0.06 | 14.19 | 31.60 | 0.16 | 12.70 |
| DEG | **17.07** | 0.07 | **9.63** | **31.00** | 0.18 | **10.56** |

### C.3 TSP and CVRP comparison in small-scale problems

To evaluate the performance of DEG on small-scale problems, we additionally conduct experiments on TSP and CVRP instances of 100 and 200 nodes. The comparisons include both diffusion-based and GFlowNet-based baselines; for CVRP, only GFlowNet-based baselines are reported because diffusion-based solvers do not directly apply to this setting. The results, summarized in Table 7, show that DEG consistently surpasses all diffusion-based and GFlowNet-based methods on TSP200, CVRP100, and CVRP200, and achieves competitive or superior performance on TSP100. These findings demonstrate that DEG also maintains strong optimization capability on problems of relatively small scales.

### C.4 Multiple Scale training

We conducted additional experiments where DEG is trained on multiple node sizes. Specifically, for both TSP and CVRP, we trained models on instances of size 100, 200, 500, and 1000, and evaluated them on larger test sets with 500, 1000, 3000, 5000, 8000, and 10000 nodes. For each training scale, we report results with and without GSA rescaling.

The results (provided in Table 8 and Table 9) show that DEG maintains strong performance even without GSA, while GSA further strengthens generalization on very large instances. These findings confirm that the scalability of DEG is not dependent on training exclusively at the 100-node scale, and that the model possesses substantial inherent scale robustness.

### C.5 Diversity Test

As shown in Table 10, we evaluate diversity on both TSP and CVRP by comparing the trajectories generated by (i) a reinforcement-learning baseline, (ii) a vanilla GFlowNet implementation (the backbone of our method), and (iii) our proposed DEG model. The diversity metric [46] is defined as the average number of edges in one trajectory that do not appear in another, a standard measure of route-level diversity in VRP. In this experiment, all settings are kept identical across models, except for the training paradigm. The results show that DEG consistently achieves higher diversity

Table 8: Training on different scales for TSP and testing without and with GSA. Obj. denotes the total route length, and Gap (%) is computed relative to LKH. For both metrics, lower is better.

| | **500** | | **1000** | | **3000** | | **5000** | | **8000** | | **10000** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Obj. | Gap(%) | Obj. | Gap(%) | Obj. | Gap(%) | Obj. | Gap(%) | Obj. | Gap(%) | Obj. | Gap(%) |
| 100(w/o GSA) | 18.54 | 12.02 | 26.74 | 15.56 | 48.45 | 20.19 | 64.97 | 25.33 | – | – | – | – |
| 200(w/o GSA) | 18.44 | 11.42 | 26.50 | 14.52 | 47.59 | 18.06 | 64.16 | 23.77 | – | – | – | – |
| 500(w/o GSA) | **18.05** | **9.06** | 26.42 | 14.17 | 47.19 | 17.07 | **61.55** | **18.73** | – | – | – | – |
| 1000(w/o GSA) | 18.41 | 11.24 | **25.87** | **11.80** | **46.32** | **14.91** | 63.14 | 21.80 | – | – | – | – |
| 100(w/ GSA) | 18.47 | 11.60 | 26.46 | 14.35 | 47.82 | 18.63 | 63.64 | 22.57 | 83.13 | 26.16 | 94.67 | 24.94 |
| 200(w/ GSA) | 18.33 | 10.76 | 26.34 | 13.83 | 46.91 | 16.37 | 62.16 | 19.91 | 82.19 | 24.74 | 91.09 | 20.20 |
| 500(w/ GSA) | **18.08** | **9.24** | 25.97 | 12.23 | 46.05 | **14.24** | **61.11** | **17.88** | **78.16** | **18.62** | **87.44** | **15.39** |
| 1000(w/ GSA) | 18.28 | 10.45 | **25.87** | **11.80** | **45.09** | 14.86 | 62.31 | 20.20 | 79.06 | 19.99 | 89.58 | 18.21 |

*"–" indicates that the model was unable to produce a result for a given instance either within the 90-minute time limit or due to GPU memory constraints.*

Table 9: Training on different scales for CVRP and testing without and with GSA. Obj. denotes the total route length, and Gap (%) is computed relative to LKH. For both metrics, lower values indicate better results.

| | **500** | | **1000** | | **3000** | | **5000** | | **8000** | | **10000** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Obj. | Gap(%) | Obj. | Gap(%) | Obj. | Gap(%) | Obj. | Gap(%) | Obj. | Gap(%) | Obj. | Gap(%) |
| 100 (w/o GSA) | 69.76 | 10.48 | 129.67 | 8.49 | 362.11 | -4.77 | 596.43 | -7.24 | 907.65 | – | 1125.04 | – |
| 200 (w/o GSA) | 67.75 | 7.30 | 127.70 | 6.84 | 362.43 | -4.69 | 580.73 | -9.68 | 987.01 | – | 1094.82 | – |
| 500 (w/o GSA) | **66.76** | **5.73** | 124.70 | 4.34 | 360.80 | -5.12 | 577.24 | -10.22 | 894.98 | – | 1072.27 | – |
| 1000(w/o GSA) | 69.68 | 10.36 | **123.13** | **3.02** | **360.42** | **-5.21** | **568.73** | **-11.54** | **885.18** | – | **1060.53** | – |
| 100 (w/ GSA) | 70.86 | 12.23 | 132.65 | 10.74 | 368.99 | -2.96 | – | – | – | – | – | – |
| 200 (w/ GSA) | 68.36 | 8.37 | 129.50 | 8.35 | 368.85 | -3.00 | – | – | – | – | – | – |
| 500 (w/ GSA) | **66.76** | **5.73** | 125.65 | 5.13 | 367.13 | -3.45 | – | – | – | – | – | – |
| 1000(w/ GSA) | 70.54 | 11.72 | **123.13** | **3.02** | **366.19** | **-3.70** | – | – | – | – | – | – |

*"–" indicates that the model was unable to produce a result for a given instance either within the 90-minute time limit or due to GPU memory constraints.*

than standard GFlowNets, demonstrating that our diffusion-enhanced design effectively enriches the exploration space.

### C.6 Learning Efficiency

We conducted additional experiments on a fixed problem scale and distribution to assess learning efficiency under controlled settings. Specifically, we used 10,000 uniformly sampled TSP and CVRP instances with size $n = 200$ and trained all models with identical configurations, differing only in the training paradigm (RL, standard GFlowNet, PO [45], and our DEG). The trained models were then evaluated on TSP/CVRP test instances at $n = 200$, $n = 500$, and $n = 1000$. The results, reported in Appendix Table 11, show that DEG exhibits strong optimization performance and superior learning efficiency compared to these baselines under matched training conditions.

### C.7 Applicability to Other Combinatorial Optimization Problems

To further assess the generalization ability of our proposed DEG beyond routing problems, we extended our experiments to additional combinatorial optimization domains. Specifically, we considered: (1) SMTWTP (Single-Machine Total Weighted Tardiness Problem), a classical single-machine scheduling problem where the objective is to minimize the total weighted tardiness of jobs processed on one machine. (2) BPP (Bin Packing Problem), a packing optimization problem that aims to minimize the number of bins required to accommodate items of varying sizes under capacity constraints, thereby improving space utilization. In Appendix Table 12, regarding SMTWTP, the instance sizes (100, 200, and 500) correspond to the number of jobs, with Obj. denoting total weighted tardiness. Regarding BPP, the instance sizes (120, 250, and 500) correspond to the number of items, with Obj. (%) denoting space utilization rate, i.e., the proportion of total bin capacity effectively used by the packed items. As suggested by those results, the proposed DEG model outperforms

Table 10: Diversity comparison on TSP and CVRP, higher values indicate better results.

| TSP | 100 | 500 | 1000 | 3000 | 5000 |
|---|---|---|---|---|---|
| RL | 17.60 | 96.07 | 217.20 | 819.34 | 1516.64 |
| GFlowNet | 20.36 | 126.95 | 294.30 | 1074.53 | 1895.23 |
| DEG | **21.79** | **128.97** | **302.63** | **1145.79** | **2082.36** |

| CVRP | 100 | 500 | 1000 | 3000 | 5000 |
|---|---|---|---|---|---|
| RL | 16.57 | 181.36 | 400.71 | 1210.16 | 2372.48 |
| GFlowNet | 19.33 | 212.27 | 443.73 | 1526.56 | 2705.91 |
| DEG | **20.54** | **217.02** | **471.68** | **1648.82** | **2909.12** |

Table 11: Comparison of training efficiency on TSP and CVRP. Obj. denotes the total route length, and Gap (%) is computed relative to LKH. For both metrics, lower values indicate better results.

| TSP | 200 | | 500 | | 1000 | |
|---|---|---|---|---|---|---|
| | Obj. | Gap(%) | Obj. | Gap(%) | Obj. | Gap(%) |
| RL | 12.12 | 14.12 | 19.28 | 16.50 | 27.82 | 20.22 |
| GFlowNet | 12.08 | 13.75 | 19.02 | 14.92 | 27.29 | 17.93 |
| PO | 11.97 | 12.71 | 18.93 | 14.38 | 27.11 | 17.16 |
| DEG | **11.91** | **12.15** | **18.79** | **13.53** | **26.80** | **15.82** |

| CVRP | 200 | | 500 | | 1000 | |
|---|---|---|---|---|---|---|
| | Obj. | Gap(%) | Obj. | Gap(%) | Obj. | Gap(%) |
| RL | 32.94 | 17.48 | 72.50 | 14.82 | 133.41 | 11.62 |
| GFlowNet | 32.03 | 14.23 | 71.49 | 13.22 | 132.47 | 10.84 |
| PO | 32.13 | 14.59 | 71.77 | 13.67 | 133.39 | 11.60 |
| DEG | **31.90** | **13.77** | **70.15** | **11.10** | **131.23** | **9.80** |

both DeepACO [61] and GFACS [27] on these tasks, demonstrating its potential as a general neural combinatorial optimization framework beyond routing.

# D Detailed Introduction of GFlowNet

The Generative Flow Network (GFlowNet) is probabilistic generative model that treats the process of sampling from a target distribution as a sequential decision-making problem [4, 5]. Formally, a GFlowNet operates on a fully-observed, deterministic Markov Decision Process (MDP) with a set of states $\mathcal{S}$, a set of actions $\mathcal{A} \subseteq \mathcal{S} \times \mathcal{S}$, and a designated initial state $s_0$. Certain states are terminal and have no outgoing actions; we denote the set of terminal states by $\mathcal{X}$. Each terminal state $x \in \mathcal{X}$ is reachable from $s_0$ through at least one (not necessarily unique) sequence of actions.

A complete trajectory is defined as $\tau = (s_0 \to s_1 \to \cdots \to s_n)$, where $s_n \in \mathcal{X}$ and every transition satisfies $(s_i, s_{i+1}) \in \mathcal{A}$. A *forward policy* $P_F(s' \mid s)$ defines the probability distribution over next states from $s$. It induces a distribution over trajectories:

$$P_F(\tau) = P_F(s_0 \to s_1 \to \cdots \to s_n) = \prod_{i=0}^{n-1} P_F(s_{i+1} \mid s_i).$$

Each terminal state $x \in \mathcal{X}$ is associated with a reward $R(x) > 0$, interpreted as an unnormalized probability mass. The learning objective of a GFlowNet is to produce a policy $P_F$ whose induced marginal distribution over terminal states satisfies

$$P_F(x) \propto R(x).$$

In other words, a GFlowNet learns to sample terminal objects with probability proportional to their reward.

**Trajectory Balance (TB).** The trajectory balance objective [39] introduces a scalar parameter $Z_\theta$ (an estimate of the partition function) and a backward policy $P_B$. For a complete trajectory $\tau = (s_0 \to \cdots \to s_n = x)$, the TB loss is

$$\ell_{\mathrm{TB}}(\tau; \theta) = \left( \log \frac{Z_\theta \prod_{i=0}^{n-1} P_F(s_{i+1} \mid s_i; \theta)}{R(x) \prod_{i=0}^{n-1} P_B(s_i \mid s_{i+1}; \theta)} \right)^2.$$

If the TB loss is driven to zero for all trajectories, the forward policy satisfies the desired reward-proportional condition. In practice, GFlowNets predict the log-probabilities of both forward and backward transitions, as well as the log-flow $F(s)$ and the log-partition constant $\log Z$.

**Training and Exploration.** TB losses depend on transitions or full trajectories, but do not prescribe how trajectories should be chosen. A common approach is on-policy training, where trajectories are sampled based on the current forward policy $P_F$ (which assigns probabilities to edges) and backward policy $P_B$ (which determines either exploration or exploitation during training), and the TB objective is minimized through gradient descent.

Table 12: Results on SMTWTP and BPP benchmarks. For SMTWTP, Obj. denotes total weighted tardiness (lower is better); for BPP, Obj. denotes space utilization rate (higher is better). Gap (%) is computed relative to LKH, where lower values indicate better performance.

| SMTWTP | 100 | | 200 | | 500 | |
|---|---|---|---|---|---|---|
| | Obj.↓ | Time(s) | Obj.↓ | Time(s) | Obj.↓ | Time(s) |
| ACO | 361.22 | 0.82 | 1608.26 | 1.70 | 11015.39 | 2.84 |
| DeepACO | 223.36 | 0.93 | 905.19 | 1.81 | 1567.20 | 3.56 |
| GFACS | 40.67 | 1.04 | 119.32 | 2.25 | 559.31 | 3.68 |
| DEG | **37.70** | **0.14** | **103.13** | **0.34** | **510.67** | **0.98** |

| BPP | 120 | | 250 | | 500 | |
|---|---|---|---|---|---|---|
| | Obj.↑ | Time(s) | Obj.↑ | Time(s) | Obj.↑ | Time(s) |
| ACO | 90.65 | 2.95 | 90.36 | 5.85 | 90.17 | 10.00 |
| DeepACO | 93.46 | 2.37 | 93.10 | 5.50 | 93.08 | 10.75 |
| GFACS | 94.68 | 2.38 | 94.03 | 5.56 | 93.65 | 10.77 |
| DEG | **96.94** | **0.27** | **96.30** | **0.58** | **95.37** | **1.38** |

**Conditional GFlowNets.** In many applications, including combinatorial optimization, the reward and admissible actions depend on an input instance. Conditional GFlowNets extend the formulation so that $P_F$ and $P_B$ are conditioned on input features, enabling generalization to unseen instances and supporting tasks where solutions correspond to variable-sized or structured objects.

## E Broad Impact and Safeguards.

Diffusion-enhanced GFlowNet for vehicle routing problems has the potential to improve the efficiency of logistics and transportation systems, reducing operational costs, fuel consumption, and environmental impact. However, applying such approaches in real-world settings requires careful testing and validation. To support reliable use, we emphasize thorough evaluation across diverse scenarios, clearly document model assumptions and limitations, and recommend combining learning-based methods with classical optimization or constraint-checking mechanisms when appropriate. Domain-specific validation is strongly encouraged to ensure responsible deployment.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction provide an overview of the key contributions and the scope of this work.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer:[Yes]

   Justification: Limitations are discussed in the conclusion section.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.

- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Theory assumptions and proofs are provided in the Methodology Section.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide the details in the Experiment Section.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.

- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The code and data will be released after publication.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We provide these details in the Experiment Section.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The results shown in the paper are average results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Experiments compute resources are shown in Experiment section.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The research comform NeurIPS Code of Ethics.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Broader impacts are discussed in Appendix D.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [Yes]

Justification: We discuss it in Appendix D.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.

- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: This paper cite the original paper properly.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We will make the code and benchmark data publicly available upon publication, with their details documented in the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper doesn't contain crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.