Graph Guided Diffusion: Unified Guidance for Conditional Graph Generation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Diffusion models have emerged as powerful generative models for graph generation, yet their use for conditional graph generation remains a fundamental challenge. In particular, guiding diffusion models on graphs under arbitrary reward signals is difficult: gradient-based methods, while powerful, are often unsuitable due to the discrete and combinatorial nature of graphs, and non-differentiable rewards further complicate gradient-based guidance. We propose Graph Guided Diffusion (GGDiff), a novel guidance framework that interprets conditional diffusion on graphs as a stochastic control problem to address this challenge. GGDiff unifies multiple guidance strategies, including gradient-based guidance (for differentiable rewards), control-based guidance (using control signals from forward reward evaluations), and zero-order approximations (bridging gradient-based and gradient-free optimization). This comprehensive, plug-and-play framework enables zero-shot guidance of pre-trained diffusion models under both differentiable and non-differentiable reward functions, adapting well-established guidance techniques to graph generation — a direction largely unexplored. Our formulation balances computational efficiency, reward alignment, and sample quality, enabling practical conditional generation across diverse reward types. We demonstrate the efficacy of GGDiff in various tasks, including constraints on graph motifs, fairness, and link prediction, achieving superior alignment with target rewards while maintaining diversity and fidelity.

1 Introduction

Diffusion models have recently shown great promise for graph generation, enabling the synthesis of realistic graph structures across diverse domains such as drug design [35], social networks [7], and molecular dynamics [10]. A key motivation behind these models is their ability to serve as *flexible generative priors*, capturing complex dependencies in both graph topology and node features. However, most existing graph diffusion models focus on unconditional or controllable generation under simple objectives. Incorporating more *general forms of rewards or constraints*, such as enforcing specific structural properties, functional motifs, or domain-specific validity criteria like fairness, remains an essential and open challenge.

Recent advances in conditional graph generation typically modify the diffusion trajectory using a conditional gradient to steer the process toward sampling from the desired conditional distribution. DiGress [33] combines a learnable regressor with classifier guidance [9], while LGD [37] adopts a similar gradient-based strategy in a latent space instead of a discrete domain like DiGress. However, these approaches require differentiable constraints, which limits their applicability in more complex graph generation tasks where constraints might be black-box functions without tractable gradients or involve discrete structures. Closer to our approach, PRODIGY [27] enforces hard constraints through

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

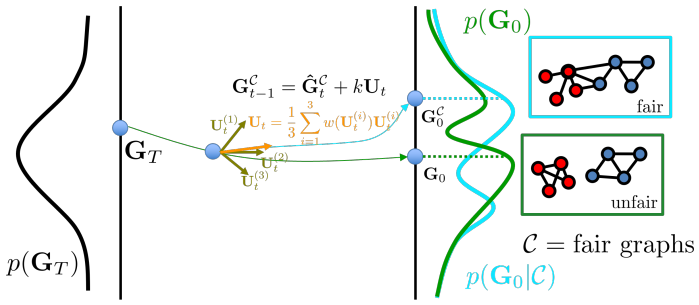


Figure 1: Illustration of GGDiff, a method that guides the generation of graphs to satisfy a set of constraints (in this case, the constraint is fairness). The guidance $\mathbf{U}_t$ is a local direction obtained via SOC, and approximated using ZO techniques, like the multi-point estimate shown here.

projected sampling along the diffusion trajectory using the bisection method [3]. This requires closed-form projection operators for a time-efficient implementation, which are often unavailable for complex constraints, necessitating expensive solvers, and limiting the applicability to more sophisticated conditions. As a result, no existing method can flexibly and effectively handle arbitrary, non-differentiable, or complex constraints in the sampling process for conditional graph generation.

In this work, we propose Graph Guided Diffusion (GGDiff), a general guidance framework for graph generation that interprets conditional graph generation as a stochastic optimal control (SOC) problem. By casting the task as a control problem, we reformulate guided diffusion as a conditional generation process with an additional control variable defined as a linear drift term. Inspired by recent advances in SOC for diffusion models [11, 23, 25], we optimize this control via path integral control, which provides an analytical yet intractable gradient. To overcome this limitation, we introduce gradient-free approximations based on zeroth-order (ZO) optimization techniques [15, 16]. Our formulation generalizes several gradient-free strategies introduced previously [11] and offers new possibilities.

GGDiff unifies various existing guidance methods in a single framework and is plug-and-play, allowing zero-shot guidance of pre-trained diffusion models under both differentiable and non-differentiable reward functions. We validate the advantages of GGDiff through extensive experiments on a wide range of constraints, including structural constraints, fairness, and link prediction. Our results demonstrate GGDiff's versatility in guiding graph generation not only towards constraints previously explored in the literature (beating current state-of-the-art architectures) but also towards arbitrary, user-defined desired outcomes (such as fair or incomplete graphs), effectively balancing precise outcome satisfaction with the preservation of the underlying graph family's characteristics.

To summarize, our contributions are threefold:

- We propose GGDiff, a framework for conditional graph generation that handles both differentiable and non-differentiable rewards by reformulating the problem as a stochastic optimal control (SOC) task.
- We introduce a general gradient-free ZO optimization formulation to handle non-differentiable rewards, enabling optimization without requiring tractable gradients.
- We conduct extensive experiments on structural, fairness, and link prediction constraints, demonstrating GGDiff's superior performance and flexibility over existing methods.

2 Controllable Generation of Graphs With General Rewards

In Section 2.1, we formulate the generation of graph conditionals as a SOC problem. Then, in Section 2.2 we propose different approximate solutions to design the control for conditional graph generation: first, in Section 2.2.1 we introduce our approximation for differentiable rewards; second, in Section 2.2.2 we propose a ZO approximation, which unifies several existing guidance policies for non-differentiable rewards. The full algorithm is provided in the appendix (Alg. 1).

2.1 Conditional Generation: A SOC Approach

The goal of our method is to *steer* a pre-trained diffusion model to sample from the posterior distribution. Importantly, we seek an algorithm that can handle general reward functions, even non-differentiable ones. To tackle this, we proposed to leverage SOC [32]. In particular, given an uncontrolled diffusion process $\mathcal{Q}$ (defined in Section A.1), we define a controlled one $\mathcal{Q}^{\mathcal{C}}$ given by

$$\mathcal{Q}^{\mathcal{C}} : \mathrm{d}\mathbf{G}_t^{\mathcal{C}} = \left[-\frac{1}{2}\mathbf{G}_t^{\mathcal{C}} - g(t)^2 \nabla_{\mathbf{G}_t^{\mathcal{C}}} \log p(\mathbf{G}_t^{\mathcal{C}}) + g(t)\mathbf{U}(\mathbf{G}_t^{\mathcal{C}}, t) \right] \mathrm{d}t + g(t)\mathrm{d}\mathbf{W}_t, \quad t \in [T, 0]. \quad (1)$$

Thus, the goal is to design the control $\{\mathbf{U}(\mathbf{G}_t^{\mathcal{C}}, t)\}_{t\in[0,T]}$ to modify the trajectory of the controlled process $\mathcal{Q}^{\mathcal{C}}$ such that the generated samples belong to the target distribution. We formalize this as a SOC problem, where we solve the following optimization problem

$$\min_{\mathbf{U}\in\mathcal{U}} \mathbb{E}\left[\int_0^T \lambda \frac{||\mathbf{U}\left(\mathbf{G}_t^{\mathcal{C}}, t\right)||_F^2}{2} \mathrm{d}t - r\left(\mathbf{G}_0^{\mathcal{C}}\right) \right] \quad \text{s.t. } \mathcal{Q}^{\mathcal{C}}. \quad (2)$$

The terminal cost in (2) represents a desired constraint for the final state $\mathbf{G}_0$ quantified by the reward $r(.)$, which is maximized (thus, the negative sign), while the transient term is a regularization term that penalizes large deviation from the uncontrolled process by promoting the energy of the controller in (1) to be small. The solution of (2) is given by the Feynman-Kac formula, a well-known result from the optimal control theory [24], given by

$$\mathbf{U}^*(\mathbf{G}_t^{\mathcal{C}}, t) = -g(t)\nabla_{\mathbf{G}_t^{\mathcal{C}}} \log \mathbb{E}_{p^{\mathrm{pre}}}\left[\exp\left(\frac{-r(\mathbf{G}_0^{\mathcal{C}})}{\lambda} \right) \Big| \mathbf{G}_t^{\mathcal{C}} \right]. \quad (3)$$

The solution in (3) is obtained as the solution of the *linear* version of the Hamilton-Jacobi-Bellman (HJB) equation [6], obtained after the exponential transformation; we deferred to Appendix B for more details on the derivation.

Given the optimal control, we now focus on how to implement it.

2.2 Estimation of the Optimal Control: A Greedy Solution

Although the expression for the optimal control derived from the Feynman-Kac formula (3) is theoretically exact, its direct computation is often intractable. Evaluating the expectation and its gradient would require simulating numerous trajectories of the uncontrolled process from the current state $\mathbf{G}_t^{\mathcal{C}}$ to the final state $\mathbf{G}_0^{\mathcal{C}}$ at each step of the generation process to estimate p^{pre}, and then backpropagating through the diffusion trajectory. This is computationally prohibitive.

We resort to a *greedy* approximation strategy to overcome this. This approach simplifies the problem by approximating the complex gradient of the log-expectation term in (3) using primarily the current state information $\mathbf{G}_t^{\mathcal{C}}$ and a one-step estimate of the clean sample $\hat{\mathbf{G}}_0^{\mathcal{C}}$. Such an approximation implies that the control decision at time t does not fully account for the entire future trajectory, potentially leading to suboptimal choices, especially in the early stages of the reverse diffusion process. However, the impact of such approximation errors may often diminish as $t \to 0$ and the state $\mathbf{G}_t^{\mathcal{C}}$ gets closer to the data. We now detail this approximation for the cases of (i) differentiable rewards and (ii) non-differentiable counterparts.

2.2.1 Differentiable Rewards

When the reward function $r(\cdot)$ is differentiable, we can derive a tractable approximation for the optimal control $\mathbf{U}^*(\mathbf{G}_t^{\mathcal{C}}, t)$.The primary challenge lies in evaluating the gradient of the log-expectation term. To circumvent this, we use Tweedie's formula (see Section A.1) to compute the MMSE denoiser $\mathbb{E}[\mathbf{G}_0^{\mathcal{C}}|\mathbf{G}_t^{\mathcal{C}}] = \hat{\mathbf{G}}_0^{\mathcal{C}}(\mathbf{G}_t^{\mathcal{C}})$ and approximate the conditional expectation in (3) as

$$\mathbb{E}_{p^{\mathrm{pre}}}\left[\exp\left(\frac{-r(\mathbf{G}_0^{\mathcal{C}})}{\lambda} \right) \Big| \hat{\mathbf{G}}_t^{\mathcal{C}} \right] \approx \exp\left(\frac{-r(\hat{\mathbf{G}}_0^{\mathcal{C}}(\hat{\mathbf{G}}_t^{\mathcal{C}}))}{\lambda} \right), \quad (4)$$

where the underlying assumption is that $p(\mathbf{G}_0^{\mathcal{C}}|\hat{\mathbf{G}}_t^{\mathcal{C}}) = \delta(\mathbf{G}_0^{\mathcal{C}} - \hat{\mathbf{G}}_0^{\mathcal{C}}(\hat{\mathbf{G}}_t^{\mathcal{C}}))$ with $\delta(.)$ denoting a Dirac delta function. This approximation becomes increasingly better as $t \to 0$ (i.e., towards the end of the reverse diffusion process), as $\hat{\mathbf{G}}_0^{\mathcal{C}}(\hat{\mathbf{G}}_t^{\mathcal{C}})$ becomes a better estimate of $\mathbf{G}_0^{\mathcal{C}}$.

Substituting this approximation into the exact optimal control formula in (3) leads to

$$\mathbf{U}^*(\hat{\mathbf{G}}_t^{\mathcal{C}}, t) \approx \frac{g(t)}{\lambda} \nabla_{\hat{\mathbf{G}}_t^{\mathcal{C}}} r(\hat{\mathbf{G}}_0^{\mathcal{C}}(\hat{\mathbf{G}}_t^{\mathcal{C}})). \tag{5}$$

This final expression provides a tractable, greedy approximation for the optimal control. The control term now directly involves the gradient of the reward function $r(\cdot)$ evaluated at the one-step denoised estimate $\hat{\mathbf{G}}_0^{\mathcal{C}}$. The term $1/\lambda$ acts as a scaling factor for the guidance. This formulation resembles guidance techniques in diffusion models, as observed by [11, 31]. For example, if the reward $r(\mathbf{G}_0)$ is proportional to the log-likelihood of a condition $\mathcal{C}$, that is, $r(\mathbf{G}_0) \propto -\log p(\mathcal{C}|\mathbf{G}_0)$, then the optimal controls boils down to the DPS approximation [5].

2.2.2 Non-differentiable Rewards

In many practical scenarios of controlled graph generation, the reward function $r(\cdot)$ is non-differentiable with respect to the generated graph $\mathbf{G}_0^{\mathcal{C}}$, rendering gradient-based approximations like (5) intractable.

To address this, we propose to determine the control input $\mathbf{U}(\mathbf{G}_t, t)$ using an approach inspired by gradient-free optimization methods [13] and ZO optimization [15]. The objective at each time t is to find a control $\mathbf{U}(\mathbf{G}_t, t)$ that steers the diffusion trajectory towards graphs yielding a high reward $r(\mathbf{G}_0^{\mathcal{C}})$. Similar to the differentiable case, we use Tweedie's formula to compute a one-step denoised version of the final graph to evaluate the reward at each time step. Given this approximation, we formally seek to find a direction $\mathbf{U}_t^*$

$$\mathbf{U}_t^* = \underset{\mathbf{U}_t}{\operatorname{argmax}}\ r\left(\hat{\mathbf{G}}_0^{\mathcal{C}}(\hat{\mathbf{G}}_t^{\mathcal{C}} + \mu \mathbf{U}_t)\right), \tag{6}$$

Here, $\hat{\mathbf{G}}_t^{\mathcal{C}} + \mu \mathbf{U}_t$ denotes the perturbed version of $\hat{\mathbf{G}}_t^{\mathcal{C}}$, which is the generated graph with the reference model at time t (before applying the guidance) following the control direction $\mathbf{U}_t$, and μ is a smoothing parameter (which depends on the noise schedule of the diffusion process). To find $\mathbf{U}_t^*$, we define a general ZO estimator for the gradient of the reward that depends on evaluations of $r(.)$ as

$$\hat{\nabla} r(\hat{\mathbf{G}}_t^{\mathcal{C}}) := \mathbb{E}_{\mathbf{U}_t \sim \mathcal{D}} \left[w(\mathbf{U}_t)\, r\left(\hat{\mathbf{G}}_0^{\mathcal{C}}(\hat{\mathbf{G}}_t^{\mathcal{C}} + \mu \mathbf{U}_t)\right) \cdot \mathbf{U}_t \right], \tag{7}$$

where $\mathcal{D}$ is a distribution over directions (typically Gaussian) and $w(\mathbf{U}_t)$ is a direction-dependent weighting function. Notably, this formulation unifies several previous gradient-free estimators. However, it is important to remark that traditional ZO optimization assumes the objective is differentiable but the gradient is inaccessible. In contrast, in our setting the reward function $r(.)$ is inherently non-differentiable, often defined via a discrete or combinatorial metric over generated graphs. Nevertheless, we treat the reward as a black-box function and employ randomized directional evaluations to define a *pseudo-gradient* direction that can guide the controlled process. Thus, the ZO estimator in (7) should be interpreted as a surrogate direction that correlates with improvements in the reward, rather than an unbiased estimator of a true gradient.

We now present three practical ZO estimators that instantiate (7).

One-point (and two-point) gradient estimators. The one-point estimator samples a single perturbation direction $\mathbf{U}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and evaluates the reward by perturbing the unconditional generated graph with this single direction. The estimated gradient is given by

$$\hat{\nabla} r(\hat{\mathbf{G}}_t^{\mathcal{C}}) = \frac{\phi(d)}{\mu}\, r\left(\hat{\mathbf{G}}_0^{\mathcal{C}}(\hat{\mathbf{G}}_t^{\mathcal{C}} + \mu \mathbf{U}_t)\right) \cdot \mathbf{U}_t, \tag{8}$$

where $\phi(d)$ is a scaling factor that depends on $\mathcal{D}$; for $\mathcal{D}$ Gaussian, we have $\phi(d) = 1$. This control corresponds to $w(\mathbf{U}_t) = \frac{\phi(d)}{\mu}$. In classical ZO, this estimator is an unbiased estimator of the smoothed version of $r(.)$ over a random perturbation, i.e., $\mathbb{E}_{\mathbf{U}_t \sim \mathcal{D}}[r(\hat{\mathbf{G}}_0^{\mathcal{C}}(\hat{\mathbf{G}}_t^{\mathcal{C}} + \mu \mathbf{U}_t))]$, but a *biased* estimator of the true reward gradient (when $\mu = 0$) and has high variance (the variance explodes as μ increases to 0) [2]. To eliminate this problem, we can use instead a two-point gradient estimator given by

$$\hat{\nabla} r(\hat{\mathbf{G}}_t^{\mathcal{C}}) = \frac{\phi(d)}{\mu} \left[r\left(\hat{\mathbf{G}}_0^{\mathcal{C}}(\hat{\mathbf{G}}_t^{\mathcal{C}} + \mu \mathbf{U}_t)\right) - r\left(\hat{\mathbf{G}}_0^{\mathcal{C}}(\hat{\mathbf{G}}_t^{\mathcal{C}})\right) \right] \cdot \mathbf{U}_t, \tag{9}$$

which is used in practice in general. For cases where $r(.)$ is differentiable, the estimator in (9) is unbiased w.r.t. true gradient (under the assumption that $\mathbb{E}_{\mathbf{U}_t \sim \mathcal{D}}[\mathbf{U}_t] = 0$ and when $\mu \to 0$.

Best-of-N direction (greedy ZO). Instead of sampling a single direction, this method samples N candidate directions $\{\mathbf{U}_t^{(1)}, \ldots, \mathbf{U}_t^{(N)}\} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and chooses the one that maximizes the reward after denoising:

$$\mathbf{U}_t^{(i)} = \underset{\{\mathbf{U}_t^{(1)}, \ldots, \mathbf{U}_t^{(N)}\}}{\operatorname{argmax}} \; r\left(\hat{\mathbf{G}}_0^{\mathcal{C}}(\hat{\mathbf{G}}_t^{\mathcal{C}} + \mu \mathbf{U}_t)\right) \cdot \mathbf{U}_t. \tag{10}$$

The final control is then set as $\mathbf{U}_t = k \cdot \mathbf{U}_t^{(i)}$, where k is a step size or scaling factor. This corresponds to using $w(\mathbf{U}_t) = \mathbb{1}(\mathbf{U}_t = \mathbf{U}_t^{(i)})$ in (7), where $\mathbb{1}$ represents the indicator function. While this method introduces bias, it often leads to effective and low-variance updates, especially when $r(\cdot)$ is highly non-smooth or sparse.

Multi-point gradient estimator (averaged random search). This variant also samples N directions $\{\mathbf{U}_t^{(1)}, \ldots, \mathbf{U}_t^{(N)}\} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, but instead of selecting the best, it forms a weighted average of all directions using their corresponding reward evaluations

$$\hat{\nabla} r(\hat{\mathbf{G}}_t^{\mathcal{C}}) = \frac{1}{N\mu} \sum_{i=1}^{N} \left[r\left(\hat{\mathbf{G}}_0^{\mathcal{C}}(\hat{\mathbf{G}}_t^{\mathcal{C}} + \mu \mathbf{U}_t^{(i)})\right) - r\left(\hat{\mathbf{G}}_0^{\mathcal{C}}(\hat{\mathbf{G}}_t^{\mathcal{C}})\right) \right] \cdot \mathbf{U}_t^{(i)}. \tag{11}$$

This approach reduces variance compared to both one-point and two-point estimators while maintaining approximate unbiasedness. It is especially useful when the reward landscape is moderately smooth, enabling the use of reward information from all sampled directions.

We defer for a quantitative analysis of variance and performance of the three estimators to Appendix C. Overall, these estimators offer flexible trade-offs between estimator quality and query complexity. In our setting, we find that the best-of-N direction yields superior performance in discrete and non-differentiable environments, typical of graph-based objectives.

3 Experiments

We evaluate the efficacy of our Graph Guided Diffusion (GGDiff) framework across several challenging tasks. We compare its three main variants—GGDiff-G (gradient-based), GGDiff-C (Best-of-N), and GGDiff-Z (multi-point)—against the state-of-the-art method PRODIGY [27] and an unconstrained baseline to highlight the impact of guidance. Our in-paper experiments cover constrained graph generation (Section 3.1), where we assess adherence to structural properties, and fair graph generation (Section 3.2), where we enforce fairness criteria. PRODIGY serves as a baseline only in the first setting, as it is unable to handle the complex reward functions required for the fairness task. A third major experiment, incomplete graph generation (link prediction), is presented in Appendix E. This appendix also contains comprehensive setup details, further use-cases and representations of the generated molecules for all experiments.

3.1 Constrained Graph Generation

We first evaluate GGDiff's performance on constrained graph generation tasks, replicating the experimental setup from the PRODIGY paper [27] to enable direct comparison. For this set of experiments, we impose constraints on the maximum degree, edge count, and maximum number of triangles of the generated graphs, on the ego small, community small, and enzymes datasets, described in Appendix E. To evaluate performance, we use two key metrics: Δ MMD, which is the metric utilized to assess PRODIGY's performance and measures the difference between the MMD values of the unconstrained dataset and the constrained generated graphs (higher values indicate that the generated graphs are closer to the original data distribution), and $\text{Val}_{\mathcal{C}}$, representing the fraction of generated graphs that successfully fulfill the imposed constraint (higher values indicate better constraint adherence).

The results for this set of experiments are presented in Table 1. They demonstrate that our GGDiff methods generally achieve superior performance compared to baselines. Specifically, GGDiff variants tend to exhibit higher Δ MMD values while also showing higher $\text{Val}_{\mathcal{C}}$ scores, demonstrating their capability to satisfy structural constraints without deviating significantly from the prior distribution of the datasets.

Table 1: Metrics comparison across datasets and constraints.

Constraint	Method	Ego Small		Community Small		Enzymes	
		Δ MMD $\uparrow$	$\text{Val}_{\mathcal{C}}$ $\uparrow$	Δ MMD $\uparrow$	$\text{Val}_{\mathcal{C}}$ $\uparrow$	Δ MMD $\uparrow$	$\text{Val}_{\mathcal{C}}$ $\uparrow$
Max Degree	GGDiff-G	0.11	0.87	-0.54	0.95	-0.37	0.98
	GGDiff-C	**0.15**	**0.90**	-0.73	**1.00**	-0.39	**1.00**
	GGDiff-Z	0.08	0.86	-0.26	0.78	-0.36	0.89
	PRODIGY	0.09	0.64	**-0.16**	0.98	**0.07**	0.95
	Uncons.	0.00	0.33	0.00	0.42	0.00	0.08
Edge Count	GGDiff-G	-0.07	**0.91**	-0.33	0.84	-0.47	**1.00**
	GGDiff-C	0.27	0.63	**-0.17**	0.91	-0.29	0.94
	GGDiff-Z	**0.28**	0.67	-0.38	0.73	-0.12	0.69
	PRODIGY	0.27	0.70	-0.39	**1.00**	**-0.10**	**1.00**
	Uncons.	0.00	0.16	0.00	0.20	0.00	0.09
Triangle Count	GGDiff-G	**0.03**	**0.96**	-0.31	0.95	-0.03	0.98
	GGDiff-C	0.01	0.89	-1.00	**1.00**	-0.01	**1.00**
	GGDiff-Z	-0.07	0.88	-0.14	0.85	-0.04	**1.00**
	PRODIGY	-0.01	0.52	**-0.13**	0.72	**0.17**	0.94
	Uncons.	0.00	0.62	0.00	0.19	0.00	0.50

3.2 Fair Graph Generation

In this section, we evaluate GGDiff's performance on generating fair graphs using metrics defined in Navarro et al. [20]. For these experiments, we randomly assign sensitive attributes to the nodes of the graphs generated from the community small dataset (for a similar experiment where the communities of the nodes are assigned by a community detection algorithm, refer to Appendix E). We report two key fairness metrics from Navarro et al. [20]: Δ DP and Δ DP_{node}, where lower values indicate greater dyadic parity and thus fairer graphs. To assess whether the generated graphs maintain the underlying SBM structure of the dataset, we report the percentage of valid SBMs. An SBM is considered valid if its estimated intra-community edge probability is at least 8 times its inter-community edge probability; this factor was chosen such that 95% of the test graphs in the dataset fulfill this criterion.

Table 2: Metrics for the fair graph generation experiment.

Method	Δ DP	$\Delta\text{DP}_{\text{node}}$	% Valid SBM
GGDiff-G	0.0026 ± 0.0029	0.0249 ± 0.0125	100.0000
GGDiff-C	0.0035 ± 0.0053	0.0192 ± 0.0121	99.2188
GGDiff-Z	0.0015 ± 0.0020	0.0061 ± 0.0037	95.3125
Uncons.	0.0071 ± 0.0145	0.0295 ± 0.0218	99.2188

The results in Table 2 demonstrate that our GGDiff methods effectively reduce the fairness metrics (Δ DP and Δ DP_{node}) and increase the number of edges between nodes with different sensitive attributes, indicating improved fairness. Crucially, these improvements are achieved while largely maintaining the generated graphs within the family of the prior distribution (SBMs), as reflected in the percentage of valid SBMs.

4 Conclusions

In this paper, we introduced **Graph Guided Diffusion (GGDiff)**, a flexible, gradient-free framework for conditional graph generation, grounded in stochastic optimal control. By casting guidance as a control problem, GGDiff enables plug-and-play conditioning of pre-trained diffusion models under both differentiable and black-box constraints. GGDiff unifies a range of existing guidance approaches, including gradient-based guidance and non-differentiable cases, under a single SOC-based formulation. Our method supports both hard and soft constraints without requiring gradient access or projection operators, making it broadly applicable across domains. Extensive experiments on structural, fairness, and topology-based constraints demonstrate GGDiff's effectiveness and generality, outperforming prior work in handling complex, non-differentiable objectives.

References

[1] Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and Van Den Berg, R. (2021). Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Inf. Process. Syst. (NeurIPS)*, 34:17981–17993.

[2] Berahas, A. S., Cao, L., Choromanski, K., and Scheinberg, K. (2022). A theoretical and empirical comparison of gradient approximations in derivative-free optimization. *Foundations of Comp. Math.*, 22(2):507–560.

[3] Boyd, S. P. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

[4] Chen, X., He, J., Han, X., and Liu, L.-P. (2023). Efficient and degree-guided graph generation via discrete diffusion modeling. *Intl. Conf. on Machine Learning (ICML)*.

[5] Chung, H., Kim, J., Mccann, M. T., Klasky, M. L., and Ye, J. C. (2022). Diffusion posterior sampling for general noisy inverse problems. In *Intl. Conf. Learn. Repr. (ICLR)*.

[6] Evans, L. C. (2022). *Partial differential equations*, volume 19. American Mathematical Society.

[7] Grover, A., Zweig, A., and Ermon, S. (2019). Graphite: Iterative generative modeling of graphs. In *Intl. Conf. on Machine Learning (ICML)*, pages 2434–2444. PMLR.

[8] Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Inf. Process. Syst. (NeurIPS)*, 33:6840–6851.

[9] Ho, J. and Salimans, T. (2021). Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*.

[10] Hoogeboom, E., Satorras, V. G., Vignac, C., and Welling, M. (2022). Equivariant diffusion for molecule generation in 3d. In *Intl. Conf. on Machine Learning (ICML)*, pages 8867–8887. PMLR.

[11] Huang, Y., Ghatare, A., Liu, Y., Hu, Z., Zhang, Q., Sastry, C. S., Gururani, S., Oore, S., and Yue, Y. (2024). Symbolic music generation with non-differentiable rule guided diffusion. *arXiv preprint arXiv:2402.14285*.

[12] Jo, J., Lee, S., and Hwang, S. J. (2022). Score-based generative modeling of graphs via the system of stochastic differential equations. In *Intl. Conf. on Machine Learning (ICML)*, pages 10362–10383. PMLR.

[13] Larson, J., Menickelly, M., and Wild, S. M. (2019). Derivative-free optimization methods. *Acta Numerica*, 28:287–404.

[14] Li, X., Zhao, Y., Wang, C., Scalia, G., Eraslan, G., Nair, S., Biancalani, T., Ji, S., Regev, A., Levine, S., et al. (2024). Derivative-free guidance in continuous and discrete diffusion models with soft value-based decoding. *arXiv preprint arXiv:2408.08252*.

[15] Liu, S., Chen, P.-Y., Kailkhura, B., Zhang, G., Hero III, A. O., and Varshney, P. K. (2020). A primer on zeroth-order optimization in signal processing and machine learning: Principals, recent advances, and applications. *IEEE Signal Process. Mag.*, 37(5):43–54.

[16] Liu, S., Kailkhura, B., Chen, P.-Y., Ting, P., Chang, S., and Amini, L. (2018). Zeroth-order stochastic variance reduction for nonconvex optimization. *Advances in Neural Inf. Process. Syst. (NeurIPS)*, 31.

[17] Luo, T., Mo, Z., and Pan, S. J. (2023). Fast graph generation via spectral diffusion. *IEEE Trans. on Patt. Analysis and Machine Int.*, 46(5):3496–3508.

[18] Madeira, M., Vignac, C., Thanou, D., and Frossard, P. (2024). Generative modelling of structurally constrained graphs. *Advances in Neural Inf. Process. Syst. (NeurIPS)*, 37:137218–137262.

[19] Minello, G., Bicciato, A., Rossi, L., Torsello, A., and Cosmo, L. (2025). Generating graphs via spectral diffusion. In *Intl. Conf. Learn. Repr. (ICLR)*.

[20] Navarro, M., Rey, S., Buciulea, A., Marques, A. G., and Segarra, S. (2024). Fair glasso: Estimating fair graphical models with unbiased statistical behavior. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C., editors, *Advances in Neural Inf. Process. Syst. (NeurIPS)*, volume 37, pages 139589–139620. Curran Associates, Inc.

[21] Niu, C., Song, Y., Song, J., Zhao, S., Grover, A., and Ermon, S. (2020). Permutation invariant graph generation via score-based generative modeling. In *Int. Conf. on Artif. Intell. and Stat.*, pages 4474–4484. PMLR.

[22] Øksendal, B. (2003). *Stochastic differential equations*. Springer.

[23] Pandey, K., Sofian, F. M., Draxler, F., Karaletsos, T., and Mandt, S. (2025). Variational control for guidance in diffusion models. *arXiv preprint arXiv:2502.03686*.

[24] Pavon, M. (1989). Stochastic control and nonequilibrium thermodynamical systems. *Applied Mathematics and Optimization*, 19:187–202.

[25] Rout, L., Chen, Y., Ruiz, N., Kumar, A., Caramanis, C., Shakkottai, S., and Chu, W.-S. (2025). Rb-modulation: Training-free personalization using stochastic optimal control. In *Intl. Conf. Learn. Repr. (ICLR)*.

[26] Rout, L., Raoof, N., Daras, G., Caramanis, C., Dimakis, A., and Shakkottai, S. (2024). Solving linear inverse problems provably via posterior sampling with latent diffusion models. *Advances in Neural Inf. Process. Syst. (NeurIPS)*, 36.

[27] Sharma, K., Kumar, S., and Trivedi, R. (2024). Diffuse, sample, project: plug-and-play controllable graph generation. In *Intl. Conf. on Machine Learning (ICML)*.

[28] Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. In *Intl. Conf. on Machine Learning (ICML)*, pages 2256–2265. PMLR.

[29] Song, J., Meng, C., and Ermon, S. (2020). Denoising diffusion implicit models. In *Intl. Conf. Learn. Repr. (ICLR)*.

[30] Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021). Score-based generative modeling through stochastic differential equations. In *Intl. Conf. Learn. Repr. (ICLR)*.

[31] Uehara, M., Zhao, Y., Wang, C., Li, X., Regev, A., Levine, S., and Biancalani, T. (2025). Inference-time alignment in diffusion models with reward-guided generation: Tutorial and review. *arXiv preprint arXiv:2501.09685*.

[32] Van Handel, R. (2007). Stochastic calculus, filtering, and stochastic control. *Course notes., URL http://www. princeton. edu/rvan/acm217/ACM217. pdf*, 14.

[33] Vignac, C., Krawczuk, I., Siraudin, A., Wang, B., Cevher, V., and Frossard, P. (2023). Digress: Discrete denoising diffusion for graph generation. In *Intl. Conf. Learn. Repr. (ICLR)*.

[34] Vincent, P. (2011). A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674.

[35] Yang, N., Wu, H., Zeng, K., Li, Y., Bao, S., and Yan, J. (2024). Molecule generation for drug design: a graph learning perspective. *Fundamental Research*.

[36] You, J., Liu, B., Ying, Z., Pande, V., and Leskovec, J. (2018). Graph convolutional policy network for goal-directed molecular graph generation. *Advances in Neural Inf. Process. Syst. (NeurIPS)*, 31.

[37] Zhou, C., Wang, X., and Zhang, M. (2024). Unifying generation and prediction on graphs with latent graph diffusion. *Advances in Neural Inf. Process. Syst. (NeurIPS)*, 37:61963–61999.

A Background and Related Works

We review graph diffusion models in the continuous domain in Section A.1, and then explain how they can be used in the context of inverse problems in Section A.2

A.1 Diffusion Models on Graphs

Diffusion models [28, 8, 30] are composed of two processes: i) a forward process that starts with clean data and gradually adds noise; and ii) a reverse process that learns to generate new data by iteratively denoising its diffused version. Graph diffusion models have been developed in both *continuous* [21, 12] and *discrete* domains [33, 4].

Discrete diffusion was introduced in DiGress [33] by adapting the structured diffusion framework [1], framing generation as edge-wise classification to mitigate combinatorial complexity. In a nutshell, the graph is treated as a multivariate categorical variable, and the diffusion process involves perturbing and recovering these discrete states. While discrete methods are well-suited for sparse graphs, they often rely on mean-field approximations and lack tractable gradients, which can limit their use in constrained generation. Given these trade-offs, we focus on the continuous setting in this work.

The continuous formulation was first introduced in EDP-GNN [21] to diffuse the graph topology, later extended in GDSS [12] to include node features, and further explored in the spectral domain [17, 19]. More recently, latent diffusion models have also been proposed [37], which operate in a learned latent space via an encoder-decoder pair. At a high level, continuous models focus on capturing *global structure*.

In this paper, we follow the formulation from GDSS [12]. We represent a graph as $\mathbf{G}_0 = \{\mathbf{X}_0, \mathbf{A}_0\}$, where $\mathbf{X}_0 \in \mathbb{R}^{N\times F}$ are node features and $\mathbf{A}_0 \in \mathbb{R}^{N\times N}$ is the weighted adjacency matrix. The forward diffusion process is defined by the stochastic differential equation $\mathrm{d}\mathbf{G}_t = -\frac{1}{2}\beta(t)\mathbf{G}_t\,\mathrm{d}t + \sqrt{\beta(t)}\,\mathrm{d}\mathbf{W}_t, \quad t \in [0, T]$, where $\beta(t)$ controls the noise schedule and is given by $\beta(t) := \beta_{\min} + (\beta_{\max} - \beta_{\min})\frac{t}{T}$. Here, $\mathbf{W}_t$ denotes standard Brownian motion. This process is designed such that the distribution of $\mathbf{G}_T$ converges to a standard Gaussian as $t \to T$. Based on this forward process, we define the reverse process as $\mathrm{d}\mathbf{G}_t = [-\frac{1}{2}\mathbf{G}_t - g(t)^2\nabla_{\mathbf{G}_t}\log p(\mathbf{G}_t)]\mathrm{d}t + g(t)\mathrm{d}\mathbf{W}_t$, where $\nabla_{\mathbf{G}_t}\log p(\mathbf{G}_t)$ is the *score function*, which is unknown, and $g(t) = \sqrt{\beta(t)}$. In particular, GDSS considers two different score functions, namely $\nabla_{\mathbf{A}_t}\log p(\mathbf{A}_t)$ and $\nabla_{\mathbf{X}_t}\log p(\mathbf{X}_t)$.

Since the true score functions are unknown, we approximate them with *score networks* $\boldsymbol{\epsilon}_{\boldsymbol{\theta}_A}(\mathbf{A}_t, t) \approx -\sigma_t\nabla_{\mathbf{A}_t}\log p(\mathbf{A}_t)$ and $\boldsymbol{\epsilon}_{\boldsymbol{\theta}_X}(\mathbf{X}_t, t) \approx -\sigma_t\nabla_{\mathbf{X}_t}\log p(\mathbf{X}_t)$, which are learned by minimizing the denoising score-matching loss [34]. After training, samples are generated using samplers like DDPM [8] and DDIM [29].

A.2 Controllable Generation of Graphs With Continuous Diffusion Models

Given a condition $\mathcal{C}$ and a reward function $r(\mathbf{G}_0)$ that quantifies how close the sample $\mathbf{G}_0$ is to meeting $\mathcal{C}$, our objective is to generate graphs that maximize this reward. From a Bayesian perspective, this problem boils down to sampling from the posterior $p(\mathbf{G}_0|\mathcal{C}) \propto p(\mathcal{C}|\mathbf{G}_0)p(\mathbf{G}_0)$, where $p(\mathcal{C}|\mathbf{G}_0) \propto \exp\left(r(\mathbf{G}_0)\right)$ is a likelihood term and $p(\mathbf{G}_0)$ is a prior given by the pre-trained diffusion model. The approaches to solving this vary significantly based on whether the reward function is differentiable.

Controllable generation with differentiable rewards. For differentiable rewards, a common strategy in inverse problems is to compute the *conditional score* using Bayes' rule: $\nabla_{\mathbf{G}_t}\log p(\mathbf{G}_t|\mathcal{C}) = \nabla_{\mathbf{G}_t}p(\mathcal{C}|\mathbf{G}_t) + \nabla_{\mathbf{G}_t}\log p(\mathbf{G}_t)$. While this allows the diffusion model to serve as a prior, the likelihood's score term is intractable. Approximations, such as using a Gaussian centered at the MMSE denoiser (computed via Tweedie's formula), have been proposed [5]. In the graph domain, methods like DiGress [33] and LGD [37] follow a similar principle, incorporating guidance via an extra, learnable model (a regressor or classifier-free guidance). However, this entire paradigm remains largely unexplored for graph inverse problems, mainly because it assumes the reward is differentiable, a condition often not met in graph generation where constraints are combinatorial [36].

Controllable generation with non-differentiable rewards. The more common case for graphs involves non-differentiable constraints. A prominent approach here is to use projection operators. PRODIGY [27], for instance, alternates between an unconditional generation step and a projection step, $\Pi_{\mathcal{C}}(\cdot)$, to enforce the constraint. While efficient for simple constraints, its applicability is severely limited by its reliance on closed-form projection operators, which are unavailable for most complex graph properties. Furthermore, applying the projection to the noisy intermediate graph $\mathbf{G}_t$ rather than the denoised estimate $\mathbb{E}[\mathbf{G}_0|\mathbf{G}_t]$ can be misaligned with the true reward domain. Other works have explored combining projection operators with edge-absorbing models [18], but these methods can be computationally demanding due to their combinatorial nature, especially when applied to discrete diffusion models.

In summary, while several methods for conditional graph generation exist, they face significant limitations. Gradient-based approaches require differentiable rewards that are rare for graphs, while projection-based methods are either restricted to simple constraints or are computationally prohibitive. This highlights the need for a more general and flexible guidance framework, which is presented in this work.

B HJB equation

In this section, we give more details on our SOC formulation. The optimal control is given by

$$\mathbf{U}^*(\mathbf{G}_t^{\mathcal{C}}, t) = -\frac{g(t)}{\lambda} \nabla_{\mathbf{G}_t^{\mathcal{C}}} V_t^* \left(\mathbf{G}_t^{\mathcal{C}}\right)$$

where $V_t^*(\mathbf{G}_t^{\mathcal{C}})$ is the optimal value function [24]. For our problem, the optimal value function at time t is given by

$$V_t^*(\mathbf{G}_t^{\mathcal{C}}) = \mathbb{E}_{p_t^*}\left[\int_t^0 \lambda \frac{\|\mathbf{U}^*(\mathbf{G}_s^{\mathcal{C}}, s)\|_2^2}{2} \mathrm{d}s - r(\mathbf{G}_0^{\mathcal{C}}) \,\middle|\, \mathbf{G}_t^{\mathcal{C}}\right] \tag{12}$$

where p_t^* denotes the optimal *controlled* distribution at time t given by $p_t^*(\mathbf{G}) \propto \exp\left(\frac{-V_t^*(\mathbf{G})}{\lambda}\right) p_t^{\mathrm{pre}}(\mathbf{G})$ and p_t^{pre} is the prior (uncontrolled) distribution[1]. The value function V_t^* solves the stochastic Hamilton-Jacobi-Bellman (HJB) equation [6], given by

$$\begin{aligned} &\partial_t V_t^*(\mathbf{G}_t^{\mathcal{C}}) = \\ &+ \left(\nabla_{\mathbf{G}_t^{\mathcal{C}}} V_t^*(\mathbf{G}_t^{\mathcal{C}})\right)^T \boldsymbol{\mu}(\mathbf{G}_t^{\mathcal{C}}, t) - \frac{g(t)^2}{2\lambda} \left\|\nabla_{\mathbf{G}_t^{\mathcal{C}}} V_t^*(\mathbf{G}_t^{\mathcal{C}})\right\|_2^2 + \frac{1}{2} g(t)^2 \Delta_{\mathbf{G}_t^{\mathcal{C}}} V_t^*(\mathbf{G}_t^{\mathcal{C}}), \end{aligned} \tag{13}$$

with boundary condition $V_0(\mathbf{G}_0^{\mathcal{C}}) = r(\mathbf{G}_0^{\mathcal{C}})$, and where $\boldsymbol{\mu}(\mathbf{G}_t^{\mathcal{C}}, t) = \frac{1}{2}\mathbf{G}_t^{\mathcal{C}} - g(t)^2 \nabla_{\mathbf{G}_t^{\mathcal{C}}} \log p(\mathbf{G}_t^{\mathcal{C}})$.

This equation is a *non-linear* partial differential equation (PDE), and the solution to the non-linear HJB equation is nontrivial. However, by applying an exponential transformation $\phi_t(\mathbf{G}_t^{\mathcal{C}}) = e^{-V_t(\mathbf{G}_t^{\mathcal{C}})}$, we can obtain the *linear* HBJ equation, given by

$$-\partial_t \phi(\mathbf{G}_t^{\mathcal{C}}, t) = \left(\nabla_{\mathbf{G}_t^{\mathcal{C}}} \phi(\mathbf{G}_t^{\mathcal{C}}, t)\right)^T \boldsymbol{\mu}(\mathbf{G}_t^{\mathcal{C}}, t) + \frac{1}{2} g(t)^2 \Delta_{\mathbf{G}_t^{\mathcal{C}}} \phi(\mathbf{G}_t^{\mathcal{C}}, t) \tag{14}$$

In particular, the Feynman-Kac formula is obtained as the solution of the linearized HJB equation in (14) (see [22] for the proof), given by

$$\exp\left(\frac{V_t^*(\mathbf{G})}{\lambda}\right) = \mathbb{E}_{p^{\mathrm{pre}}}\left[\exp\left(\frac{-r(\mathbf{G}_0^{\mathcal{C}})}{\lambda}\right) \,\middle|\, \mathbf{G}_t^{\mathcal{C}} = \mathbf{G}\right]. \tag{15}$$

This leads to an expression for the optimal control in terms of the reward function as given by (3).

Stochastic optimal control for zero-shot controlled generation. Recent methods have proposed the use of SOC for controlled generation [31, 14]. In the context of music generation [11], the authors propose a method to generate samples when likelihoods are non-differentiable. In [26], a linear quadratic control was proposed for style transfer in image generation. More recently, a non-linear control formulation was introduced in [23] for image inverse problems. However, as far as we are concerned, the application of SOC for graph generation has not been explored yet.

[1] We assume here that the terminal time is 0 and the time runs backwards (so $t < 0$).

C Background on zeroth-order optimization

In Section 2.2.2, we leverage zeroth-order optimization for defining a surrogate gradient of the reward function. We propose three estimators in particular, where each one has its own properties. In this section, we expand on them.

Two-point gradient estimator. The two-point gradient estimator in (9) is the first one that we introduced. This estimator has a mean-squared error given by

$$\mathbb{E}[\|\hat{\nabla} r(\mathbf{G}_0) - \nabla r((\mathbf{G}_0))\|_2^2] = O(d)\|\nabla r(\mathbf{G}_0)\|_2^2 + O\left(\frac{\mu^2 d^3 + \mu^2 d}{\phi(d)}\right) \tag{16}$$

The proof can be found in [16]. The error in (16) sheds light on the behavior of this estimator. First, the second term depends on the parameter $\boldsymbol{\mu}$: when this parameter gets smaller, the gradient estimate gets better. However, if $\boldsymbol{\mu}$ becomes too small, then the effect of the guidance diminishes. Second, the first term depends on the dimension d. This imposes a variance which cannot be 0 even for small values of μ.

Multi-point gradient estimator. The third estimator is based on the multi-point gradient estimate, which computes an average over random directions. This estimator has a mean-squared error given by

$$\mathbb{E}[\|\hat{\nabla} r(\mathbf{G}_0) - \nabla r((\mathbf{G}_0))\|_2^2] = O\left(\frac{d}{N}\right)\|\nabla f(\mathbf{x})\|_2^2 + O\left(\frac{\mu^2 d^3}{\phi(d)N}\right) + O\left(\frac{\mu^2 d}{\phi(d)}\right) \tag{17}$$

Compared to the two-point case, the error in (17) depends on the number of samples that are used to compute the average. In particular, the first two terms go to 0 when $N \to \infty$; the third term is independent of N, and corresponds to the approximation error between the true gradient and the smoothed version. However, it is controlled by the smoothing parameter $\boldsymbol{\mu}$.

A summary of each estimator is shown in Table 3.

Table 3: Comparison of ZO estimators for control direction optimization.

Method	**Variance**	**Reward evaluation**
2-Point Estimator	High	2
Best-of-N Direction	Low	N
Averaged Random Search	Moderate	$N+1$

D Final algorithm

We put everything together and show our proposed algorithm in Alg. 1.

E Experimental Details

This appendix provides detailed information regarding the experimental setup used in this paper, including specifics about the datasets, computational resources utilized, and a comprehensive description of additional experiments conducted. The appendix is structured as follows: Section E.1 includes a description of the datasets used for evaluating GGDiff's performance. Section E.2, details the computational resources of the server where the experiments were run. Section E.3 presents additional experimental results, with subsections dedicated to further details on constrained graph generation (Section E.3.1), fair graph generation (Section E.3.2) and link prediction (Section E.3.3).

E.1 Datasets

We evaluate our proposed GGDiff framework and baselines on a selection of benchmark graph datasets, encompassing both generic network structures and molecular graphs. The datasets used in our experiments are described below:

Algorithm 1 GGDiff for controllable generation on graphs

Require: $T, \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{G}_t, t), N, k, \mu, \{\alpha_t\}_{t=0}^{T}, \{\sigma_t\}_{t=0}^{T}, r(\cdot)$

Sample $\mathbf{G}_T^{\mathcal{C}}$ from $p(\mathbf{G}_T)$.
for $t = T-1$ to 1 **do**
  $\hat{\mathbf{G}}_t^{\mathcal{C}} = \frac{1}{\sqrt{\alpha_{t+1}}}\left(\mathbf{G}_{t+1}^{\mathcal{C}} - \frac{1-\alpha_{t+1}}{\sqrt{1-\bar{\alpha}_{t+1}}}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{G}_{t+1}^{\mathcal{C}}, t+1)\right)$ (DDPM update).
  if r is differentiable **then**
    Compute $\hat{\mathbf{G}}_0^{\mathcal{C}}(\hat{\mathbf{G}}_t^{\mathcal{C}}) = \frac{1}{\alpha_t}\left(\hat{\mathbf{G}}_t^{\mathcal{C}} + \sigma_t^2 \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\hat{\mathbf{G}}_t^{\mathcal{C}}, t)\right)$.
    Compute $\mathbf{U}_t = \nabla_{\hat{\mathbf{G}}_t^{\mathcal{C}}} r(\hat{\mathbf{G}}_0^{\mathcal{C}}(\hat{\mathbf{G}}_t^{\mathcal{C}}))$ using (5).
  else
    Sample N candidates $\{\mathbf{U}_t^{(1)}, \ldots, \mathbf{U}_t^{(N)}\} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
    Compute $\tilde{\mathbf{G}}_t^{\mathcal{C},(i)} = \hat{\mathbf{G}}_t^{\mathcal{C}} + k\mathbf{U}_t^{(i)}$ for $i = 1, \cdots, N$.
    Compute $\hat{\mathbf{G}}_0^{\mathcal{C},(i)} = \frac{1}{\alpha_t}\left(\tilde{\mathbf{G}}_t^{\mathcal{C},(i)} + \sigma_t^2 \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\tilde{\mathbf{G}}_t^{\mathcal{C},(i)}, t)\right)$ for $i = 1, \cdots, N$.
    Approximate $\hat{\nabla} r(\hat{\mathbf{G}}_t^{\mathcal{C}})$ using (9), (10), (11)
    if Approximation of $\hat{\nabla} r(\hat{\mathbf{G}}_t^{\mathcal{C}})$ is (10) **then**
      Set $\mathbf{U}_t = \operatorname{argmax}_{\mathbf{U}_t^{(i)}} r(\hat{\mathbf{G}}_0^{\mathcal{C},(i)})$.
    else if Approximation of $\hat{\nabla} r(\hat{\mathbf{G}}_t^{\mathcal{C}})$ is (9) or (11) **then**
      Set $\mathbf{U}_t = \hat{\nabla} r(\hat{\mathbf{G}}_t^{\mathcal{C}})$.
    end if
  end if
  $\mathbf{G}_t^{\mathcal{C}} = \hat{\mathbf{G}}_t^{\mathcal{C}} + k\mathbf{U}_t$.
end for
return $\mathbf{G}_0^{\mathcal{C}}$

1. **Ego-small**: This dataset comprises 200 small ego graphs extracted from the larger Citeseer network.
2. **Community-small**: Consisting of 100 synthetic graphs, this dataset features structures exhibiting distinct community partitions.
3. **Enzymes**: We use the protein graphs from the BRENDA enzyme database, totaling 587 graphs.
4. **QM9**: A molecular dataset containing approximately 133,000 small molecules. These molecules are composed of up to 9 heavy atoms, including Carbon (C), Nitrogen (N), Oxygen (O), and Fluorine (F).
5. **ZINC250k**: This large molecular dataset includes 250,000 drug-like molecules. The graphs represent molecules with 6 to 38 heavy atoms, incorporating Carbon (C), Nitrogen (N), Oxygen (O), Fluorine (F), Phosphorus (P), Chlorine (Cl), Bromine (Br), and Iodine (I).

E.2 Computational resources

All experiments were conducted on a server equipped with an AMD EPYC 9634 84-Core Processor and 512GB of total physical memory (RAM). For accelerated computation, the server uses an NVIDIA GeForce RTX 4090 graphics processing units (GPUs), each featuring 24GB of dedicated video memory. The software environment runs on Ubuntu 24.04 LTS, with NVIDIA driver version 560.35.03 and CUDA version 12.6.

E.3 Additional experiments

E.3.1 Constrained Graph Generation

In this section, we provide additional details regarding the constrained graph generation experiments summarized in the main paper (see Table 1). For comparison purposes with prior work, we specifically focus on evaluating GGDiff's performance on the task of guiding the generated graphs towards fulfilling the constraints previously defined and utilized in Sharma et al. [27]. These constraints, designed to enforce specific structural properties, are presented in Table 4, along with their descriptions and mathematical formulations.

Table 4: Summary of Constraints from [27]

Constraint Type	Limiting factor	Mathematical Formulation
Edge Count	Number of edges $\lvert\mathcal{E}\rvert$	$\lvert\mathcal{E}\rvert = \mathbf{1}^\top \mathbf{A}\mathbf{1} \leq B$ for a given constant $B \geq 0$
Triangle Count	Number of triangles	$\mathrm{tr}(A^3) \leq T$ for a given constant $T \geq 0$
Degree	Maximum Degree	$\max_i [\mathbf{A}\mathbf{1}]_i \leq D$ for a given constant D

Table 5: Metrics for the force stars constraint in the Ego small dataset.

Method	% 1 Node	% Stars	% Stars & > 1 Node	% Valid Egonet	Edges over Star
GGDiff-G	0.78	53.12	52.34	96.09	1.08 ± 2.61
GGDiff-L	2.34	51.56	49.22	88.28	0.44 ± 0.58
PRODIGY	100.00	100.00	0.00	100.00	0.00 ± 0.00
Uncons.	0.78	24.22	23.44	99.22	1.86 ± 2.64

The values for constants B, T, and D used for each dataset are selected based on those reported in [27] to ensure a direct comparison of method performance under identical constraint settings, and are given by those values fulfilled by 10% of the graphs in the test dataset.

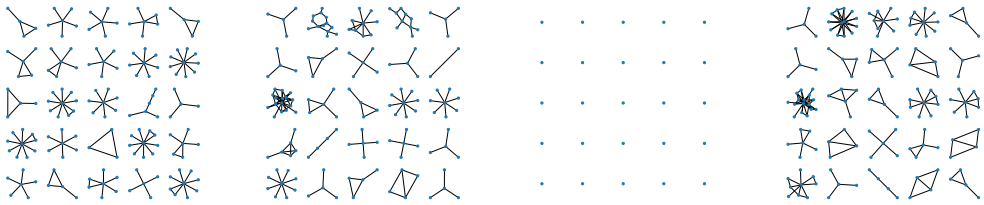
The specific loss function used for each constraint is empirically selected from a pool of possibilities based on which yields the best performance; a comprehensive list of options can be found in the code associated with this submission. For differentiable guidance (Section 2.2.1), the choice is restricted to differentiable functions, typically involving ℓ_1 or ℓ_2 norms. For instance, an ℓ_2 loss for the edge count constraint could be $(\mathbf{1}^\top \hat{\mathbf{A}}_0^{\mathcal{C}}(\mathbf{A}_t^{\mathcal{C}})\mathbf{1} - B)^2$. In contrast, the non-differentiable (zero-order) guidance (Section 2.2.2) significantly expands the available loss functions. Examples include utilizing non-differentiable operations in the differentiable losses, like using the quantized adjacency via the entry-wise indicator function $\mathbb{1}(\hat{\mathbf{A}}_0^{\mathcal{C}}(\mathbf{A}_t^{\mathcal{C}}) > 0.5)$ in lieu of the estimate $\hat{\mathbf{A}}_0^{\mathcal{C}}(\mathbf{A}_t^{\mathcal{C}})$, or employing one-sided penalties such as $\max\{\mathbf{1}^\top \hat{\mathbf{A}}_0^{\mathcal{C}}(\mathbf{A}_t^{\mathcal{C}})\mathbf{1} - B, 0\}$.

Moving beyond the constraints explored in Sharma et al. [27], we investigate GGDiff's ability to generate star graphs within the Ego small dataset. As directly enforcing a star graph structure is outside the standard constraints that can be achieved via projection, for PRODIGY we proxy this by setting the number of triangles to 0, a necessary condition for star graphs. The results are detailed in Table 5, where we report the percentage of graphs with 1 node, the percentage of generated graphs that are stars, the percentage of stars with more than one node, the percentage of valid egonets, and the difference in the number of edges with respect to a star graph, i.e., the ratio between the number of generated graphs that fulfill the condition (having one node, valid egonet, etc.) and the total number of generated graphs. Our findings indicate that PRODIGY generates graphs consisting of only a single node, as it can be appreciated in Figure 2. In contrast, our GGDiff methods successfully double the percentage of generated star graphs compared to the unconstrained case, while effectively preserving the overall data distribution, as approximately 90% of the graphs generated by GGDiff are valid egonets. Notice that all graphs generated by PRODIGY are valid egonets because a graph with a single node is considered a valid egonet. Additionally, our methods substantially reduce the number of excess edges beyond what is required for a star graph over the unconstrained case.

E.3.2 Fair graph generation

In this appendix section, we provide further details regarding the fair graph generation experiments introduced in the main paper. These experiments evaluate GGDiff's ability to generate graphs that satisfy fairness criteria based on assigned sensitive attributes. To encourage fair graphs, we employ the same loss functions defined in Navarro et al. [20]. We investigate two distinct methods for assigning sensitive attributes to the nodes of the community small dataset:

1. **Random assignment**: Sensitive attributes are assigned to nodes randomly. The quantitative results for the fairness metrics and SBM validity for this scenario are presented in Table 2 in the main paper.
2. **Community partitioning algorithm-based assignment**: Sensitive attributes are assigned to nodes based on the community structure identified by a community partitioning algorithm. This represents a more challenging scenario for generating fair graphs that are also valid



(a) GGDiff-C. (b) GGDiff-G. (c) PRODIGY. (d) Unconstrained.

Figure 2: Samples for the force stars constraint in the Ego small dataset.

Stochastic Block Models (SBMs). Since SBMs are characterized by a high density of intra-community edges and a low density of inter-community edges, aligning the sensitive attribute with community membership creates a direct tension: the fair loss function encourages the formation of edges between nodes with different attributes (i.e., nodes in different communities), while the underlying data distribution and the objective of generating valid SBMs favor the opposite.

Analyzing the results for the community partitioning-based assignment presented in Table 6, our three GGDiff methods are still able to effectively reduce the fairness metrics compared to baselines, while largely maintaining a high percentage of valid SBMs. This quantitative improvement is visually corroborated by the sample graphs shown in Figure 3, where graphs generated by GGDiff show a higher density of edges connecting nodes of different sensitive attributes (indicated by node color) compared to the unconstrained case.

Table 6: Metrics for the fair graph generation with community partition.

Method	**Δ DP**	**ΔDP$_{\text{node}}$**	**% Valid SBM**
Greedy	0.3389 ± 0.0763	0.1931 ± 0.0387	98.4375
Loss	0.3119 ± 0.1289	0.1892 ± 0.0720	77.3438
Zero	0.3451 ± 0.0612	0.1999 ± 0.0498	98.4375
Uncons	0.4133 ± 0.0769	0.2348 ± 0.0449	99.2188

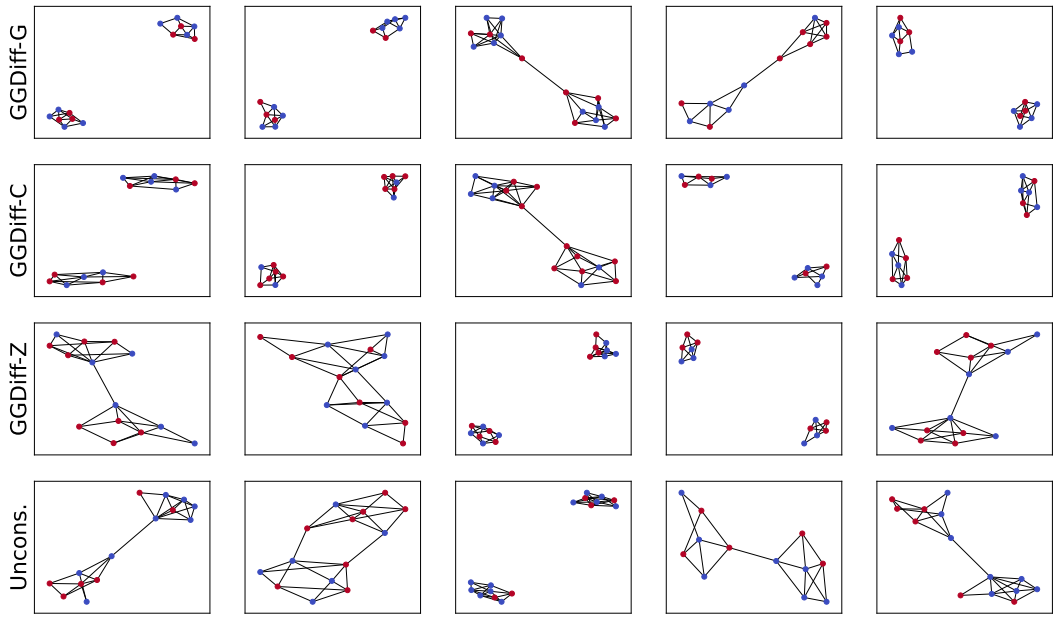
E.3.3 Incomplete graph generation

In this task, we evaluate GGDiff's ability to generate graphs consistent with partially observed adjacency matrices. Specifically, we assume that 50% of the entries of the adjacency matrix are observed and should be maintained in the generated graph. We conduct these experiments on two molecular datasets, QM9 and ZINC250k. We evaluate performance using three metrics: Accuracy, which measures the percentage of observed entries that are respected in the generated graphs; and % Unique, the percentage of generated molecules that are novel compared to the training set.

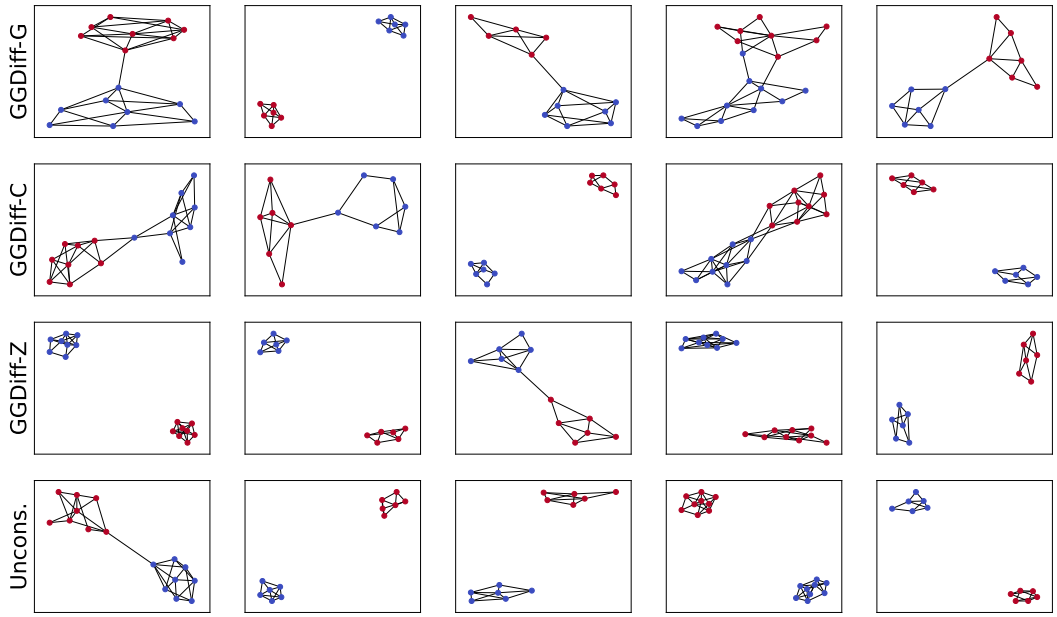
The results are presented in Table 7. Our GGDiff methods, particularly GGDiff-G and GGDiff-Z, demonstrate high accuracy in respecting the observed entries, both of them achieving almost 90% accuracy on QM9 and over 98% accuracy on ZINC250k. The unconstrained case also shows relatively high accuracy, which is largely attributable to the high prevalence of zero entries (absence of edges) in sparse graphs. For a more challenging evaluation where we observe edges instead of entries, please refer to Appendix E. Across all methods, the percentage of valid generated molecules is consistently 100%, likely aided by the partial observation of the adjacency matrix. We observe a trade-off between

Table 7: Results for the incomplete graph generation experiment.

Method	**QM9**		**ZINC250k**	
	Acc. (%)	% Unique	Acc. (%)	% Unique
GGDiff-G	91.39	73.57	98.85	100.00
GGDiff-C	67.20	94.32	95.27	100.00
GGDiff-Z	88.72	90.88	98.44	100.00
Uncons.	61.51	97.86	93.43	100.00



(a) Random community assignment.



(b) Community partitioning algorithm-based assignment.

Figure 3: Samples from the fair graph generation experiment.

accuracy and novelty in the QM9 dataset: as the accuracy in fixing observed entries increases, the percentage of novel molecules tends to decrease, suggesting that achieving very high fidelity to observed structure can lead to generating molecules highly similar to those in the test set. This tradeoff isn't observed in the ZINC250k dataset, likely due to the fact that the graphs are larger and therefore the model has more freedom to adapt to the observed entries.

This section provides additional details on the link prediction experiments, also referred to as incomplete graph generation. In this task, we evaluate GGDiff's ability to generate graphs where a subset of adjacency matrix entries is observed and must be precisely replicated in the generated output. We investigate two scenarios for the observed entries: (i) observing a random 50% of all adjacency matrix entries (both existing edges and non-edges), and (ii) observing only a random subset of existing edges (entries equal to 1). This task is particularly relevant in domains like molecule generation, where there is often an interest in generating molecules that incorporate a specific predefined substructure (e.g., a benzene ring). Enforcing observed edges allows for the generation of molecules that respect such topological constraints.

The results for the first scenario (observing random entries) are presented in Table 7 in the main paper. As noted, the high accuracy values observed in this case are significantly influenced by the correct generation of prevalent zero entries (non-edges) that were part of the observed subset. We now detail the second, more challenging scenario in this appendix.

For the second scenario, we observe only a random subset of existing edges in the adjacency matrix. The results for this case are presented in Table 8. Here, the accuracy metric specifically measures how well the generated graphs reproduce the observed edges. As expected, the accuracy values drop significantly compared to the first scenario because correctly generating existing edges is a more stringent condition than correctly generating non-edges in sparse graphs. However, the effect of GGDiff's guidance becomes strikingly apparent: our methods, particularly GGDiff-G and GGDiff-Z, achieve drastically increased accuracy in reproducing the observed edges on both the QM9 and ZINC250k datasets compared to the unconstrained baseline.

Sample graphs illustrating the results of the link prediction experiment are shown in Figure 4. In these visualizations, we use color and line style to indicate the status of observed entries in the generated graphs:

- **Solid green lines**: Observed edges that were successfully preserved in the generated graphs.
- **Solid red lines**: Observed edges that were *not* preserved in the generated graphs.
- **Dotted green lines**: Observed non-edge entries that were correctly preserved as non-edges.
- **Dotted red lines**: Observed non-edge entries that were *not* preserved (i.e., incorrectly generated as edges).

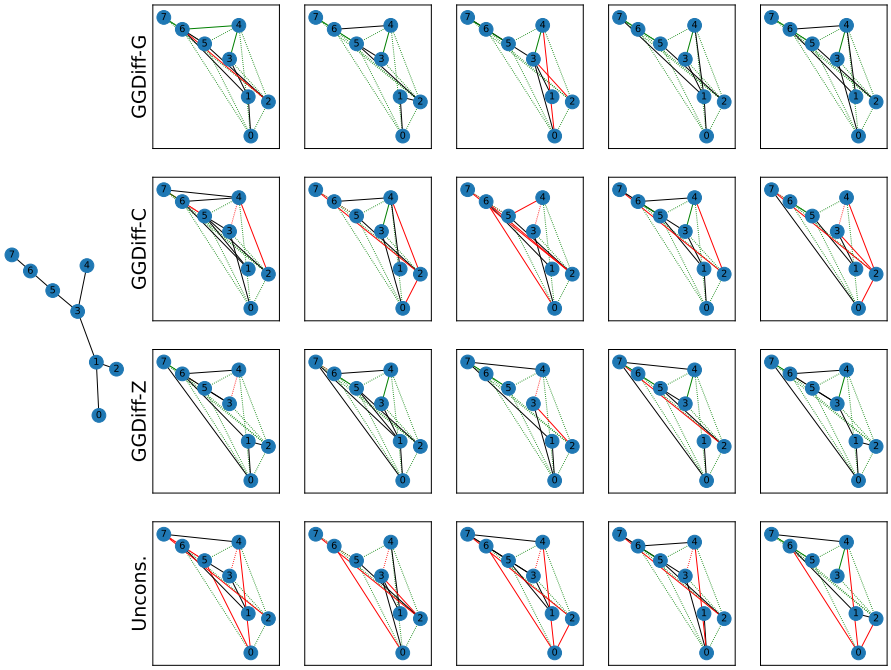
As observed in the figure, graphs generated by our GGDiff methods exhibit a clear prevalence of green lines (indicating high preservation of observed entries and edges), whereas the unconstrained case shows a greater number of red lines, highlighting its inability to reliably reproduce the specified topological constraints.

Table 8: Results for the incomplete graph generation experiment with observed edges.

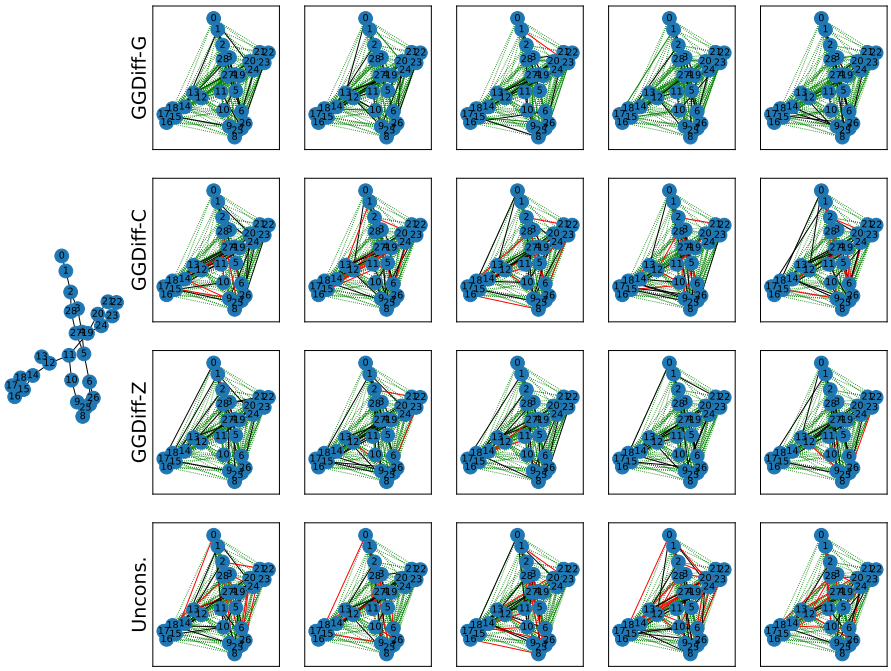
Method	**QM9**		**ZINC250k**	
	Acc. (%)	% Unique	Acc. (%)	% Unique
GGDiff-G	79.73	86.53	95.98	100.00
GGDiff-C	29.18	97.56	19.73	99.90
GGDiff-Z	41.66	92.71	85.59	100.00
Uncons.	23.40	97.81	8.45	100.00

F Social impacts

The generation of graphs under constraints could lead to undesired consequences if not applied with care. For example, when doing graph completion. In sensitive applications like healthcare or finance,



(a) QM9.



(b) ZINC250k.

Figure 4: Samples generated for the incomplete graph generation experiment. The graph on the left is the test graph from which we observe the entries in its adjacency matrix. The generated graphs are represented in the rows, one for each of the methods. In the generated graphs, the solid green (red) lines are observed edges that were (not) preserved in the generated graphs, while dotted green (red) lines are observed entries not corresponding to an edge that were (not) preserved in the generated graphs.

these prediction inaccuracies can have serious repercussions, including misdiagnosis or financial losses. Moreover, misusing these models in social network analysis might inadvertently reinforce biases or invade privacy if not handled ethically. Therefore, it is crucial to apply GGDiff and any other graph inference algorithm using diffusion models with a thorough understanding of their limitations and to validate results rigorously to mitigate these risks.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

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 reflects the paper's content. We claim that we propose a framework for controllable generation using diffusion models for non-differentiable constraints. In particular, we define the guidance via stochastic optimal control, and we approximate the optimal control using zeroth-order optimization. We support the benefits of our algorithm with comparisons against other benchmarks and with ablation studies.

   Guidelines:

   - The answer NA 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 NA 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: See Section 4

Guidelines:

- The answer NA 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: We provide all the necessary proofs in the Appendix. And when we do not provide, we cite the corresponding work.

Guidelines:

- The answer NA 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: Our algorithm is clearly described in Algorithm 1, and all the hyperparameters and datasets used are explicitly reported (see Section 3 and Appendix E). The source code is also provided in the submission (it will be publicly available on GitHub if the paper gets accepted).

Guidelines:

- The answer NA 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: [Yes]

Justification: Out supplementary material includes both Appendix E where the details for the experimental evaluation are provided, as well as the code used for the experimental results, with the hyperparameters clearly organized in YAML files.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.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://nips.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, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We describe the setup and all the details in Section 3 and Appendix E.

   Guidelines:

   - The answer NA 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 standard deviations are provided in the corresponding tables whenever possible.

   Guidelines:

   - The answer NA 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: The computer resources are included in the supplementary material, more precisely in Appendix E.2.

Guidelines:

- The answer NA 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: We have read the NeurIPS Code of Ethics and ensured the paper satisfies every aspect.

Guidelines:

- The answer NA 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: We discuss the societal impact in Appendix F.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our paper do not pose any risk.

Guidelines:

- The answer NA 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: The only assets that are not ours and were used in this work are the code for the unconditional diffusion model (credited when citing the original papers), and the public datasets (credited in Section 3).

Guidelines:

- The answer NA 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: Well-documented source code is submitted and will be available on GitHub if the paper is accepted.

Guidelines:

- The answer NA 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: [NA]

    Justification: We did not do experiments with human subjects.

    Guidelines:

    - The answer NA 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: [NA]

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

    Guidelines:

    - The answer NA 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 rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: Our method does not involve LLMs in any of the core modules.

    Guidelines:

    - The answer NA 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 (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.