GRAFT: RELATIONAL NODE FIELDS FOR LATENT GRAPH GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph generation is challenging because structural relations are inherently pairwise, leading many diffusion and flow models to maintain $\mathcal{O}(n^2)$ adjacency, edge, or pair-level states throughout generation. Latent graph generation offers a more compact alternative, but existing approaches often lack relational representations rich enough to preserve graph structure, limiting their ability to realize the scalability benefits of node-only transport as graph size grows. We propose GRAFT, a latent graph generation framework that constructs relational node fields and transports them using Flow Matching. GRAFT folds pairwise and global relational information into node representations while keeping the entire generative trajectory node-only, with self-conditioning used to improve node-field transport. GRAFT achieves competitive generation quality across diverse graph domains while substantially improving sampling efficiency. On Protein, GRAFT samples a graph in 0.290 s with 110.6 MiB of peak GPU memory under each method's native inference configuration, i.e. $2.1\times$ to $148.7\times$ faster and $1.9\times$ to $5.7\times$ lower in peak memory than the evaluated methods. Code is available at `https://anonymous.4open.science/r/GRAFT-D1A0`.

1 INTRODUCTION

Diffusion models and Flow Matching have achieved strong performance in computer vision and natural language processing, and have recently been extended to graph generation. Representative examples include discrete diffusion models such as DiGress (Vignac et al., 2023), continuous-time formulations such as DisCo (Xu et al., 2024) and Cometh (Siraudin et al., 2024), and flow-based methods such as DeFoG (Qin et al., 2025). Graphs, however, differ fundamentally from images and sequences: their structure is defined by relations between pairs of nodes. Consequently, many graph generative models explicitly maintain adjacency matrices, edge variables, or pair-level representations throughout the generative trajectory. These pair-level states scale as $\mathcal{O}(n^2)$ and must be repeatedly stored and updated at every diffusion or flow step, leading to rapidly increasing memory and runtime costs as graph size grows.

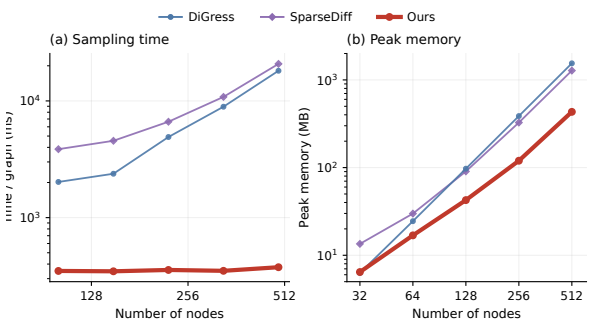


Figure 1: Protein scalability. (a) At 100 NFEs, GRAFT maintains a substantial sampling-time advantage across the evaluated graph sizes. (b) Under matched 100-NFE inference, GRAFT uses the lowest peak GPU memory, with a larger advantage on larger graphs.

A natural alternative is to generate graphs in a node-level latent space, reducing the transported state from $\mathcal{O}(n^2)$ pairwise variables to n node representations. Recent latent graph generators such as LG-Flow (Siraudin & Morris, 2026) have shown that such node-only transport is feasible. However, their latent construction remains largely node-centric: the encoder combines Laplacian positional encodings (Dwivedi & Bresson, 2020) with message passing along observed edges, without maintaining dense pair states for non-adjacent or multi-hop relations or any graph-level context (Siraudin & Morris, 2026). As we show empirically, this becomes problematic on structurally challenging

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

benchmarks such as SBM (Martinkus et al., 2022) and Protein (Bergmeister et al., 2024), where node-centric latent construction fails to retain sufficient relational information for accurate reconstruction. This raises two questions: how should a relational node field be constructed, and how can it be reliably transported while keeping the generative trajectory entirely node-only?

We propose **GRAFT**, a latent graph generation framework that addresses both problems. GRAFT is built on the observation that removing pair states from the transported state does not remove relational dependencies: a node field must still preserve the relations required for graph realization. We construct a relational node field by incorporating pairwise and graph-level information before compression, while discarding these states before generation. We then transport this relational node field using Flow Matching (Lipman et al., 2023; Liu et al., 2023) while keeping the entire generative trajectory node-only. We further analyze the sensitivity of relational decoding and use self-conditioning (Chen et al., 2022) to reduce transport-error accumulation before decoding.

Our study follows the two components of GRAFT. For relational field construction, GRAFT achieves 100% reconstruction on MOSES (Polykovskiy et al., 2020), GuacaMol (Brown et al., 2019), and Planar (Martinkus et al., 2022), 96.88% on SBM (Martinkus et al., 2022), and 93.75% on Protein (Bergmeister et al., 2024). For node-field transport, we theoretically characterize relational error amplification: although the transported state contains only node variables, transport perturbations can induce amplified relational errors after decoding. We empirically verify this phenomenon on actual transport trajectories and use self-conditioning (Chen et al., 2022) to reduce the transport error exposed to relational decoding. Together, GRAFT preserves relational structure while keeping the entire generative trajectory node-only. The efficiency advantage becomes increasingly pronounced as graph size grows: under controlled matched-NFE inference, GRAFT exhibits substantially better runtime and memory scaling than representative pair-state generators (Vignac et al., 2023; Chen et al., 2025) (Fig. 1).

Our contributions are threefold:

- We formulate latent graph generation as the problem of **constructing and transporting relational node fields**: graph relations may be removed from the transported state, but their dependencies must remain encoded in the joint node representation.
- We propose **GRAFT**, which combines relational field construction with Flow Matching over the node field. Relational reasoning is confined to field construction and final graph realization, while the entire generative trajectory remains node-only.
- We analyze relational sensitivity in node-only graph generation and show that transport perturbations can lead to amplified relational errors after decoding. We further verify this trend on actual Flow transport trajectories and evaluate self-conditioning as a practical mechanism for reducing transport error before relational decoding.

2 Related Work

Diffusion and flow matching for graph generation. Graph-space generative models typically evolve explicit node and edge variables throughout the generation trajectory (Martinkus et al., 2022; Minello et al., 2025). Representative approaches include continuous-state diffusion such as GDSS (Jo et al., 2022), discrete denoising in DiGress (Vignac et al., 2023), continuous-time discrete-state models such as DisCo (Xu et al., 2024) and Cometh (Siraudin et al., 2024), and discrete flow matching in DeFoG (Qin et al., 2025). Despite differences in their generative dynamics, these methods explicitly maintain candidate node relations during sampling, resulting in an $\mathcal{O}(n^2)$ pair-level state at fixed feature width. Sparse approaches such as EDGE (Chen et al., 2023) reduce the number of relations processed or generated, but still operate on explicit graph structure. In contrast, our approach removes pair-level variables from the transported state entirely: relational reasoning is performed when constructing the node field, while Flow Matching evolves only n node representations. The transported state therefore scales as $\mathcal{O}(nd)$, although pairwise computation may still appear during field construction and final graph realization.

Latent graph generation. Latent methods move graph generation into a compact representation to avoid transporting explicit $\mathcal{O}(n^2)$ pair-level states (Fu et al., 2024). LGD (Zhou et al., 2024) still generates augmented edge latents, while GLAD (Nguyen et al., 2025) uses node latents but

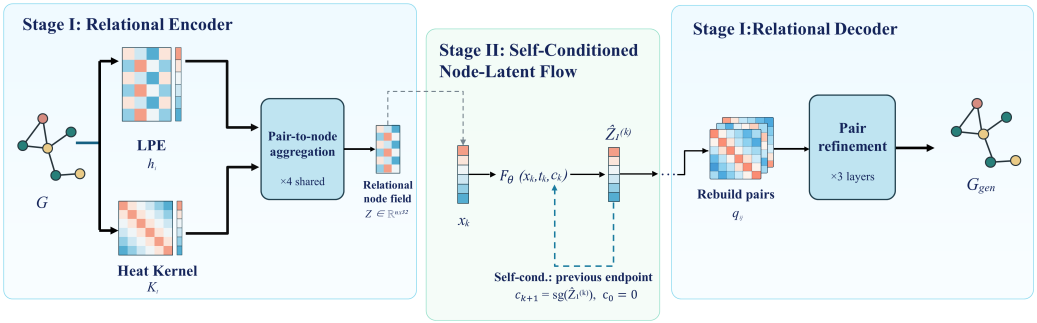


Figure 2: **Overview of GRAFT.** Stage I constructs the relational node field: a relational encoder aggregates transient pair states, built from node features, Laplacian positional encodings, and heat-kernel descriptors, into a fixed-width node field, and a mirrored relational decoder re-instantiates pair states from that field to realize the graph. Stage II freezes Stage I and transports only the node field with a self-conditioned Flow Matching model, so no pair-level state is generated during transport.

retains all-pairs computation in its generative backbone. LG-Flow (Siraudin & Morris, 2026) takes a fully node-only route by constructing fixed-width node embeddings with Laplacian positional encodings and message passing. GRAFT retains this node-only transport principle but changes how the field is constructed: it explicitly aggregates transient pairwise and global relational information before compression. In our matched Stage-I comparison, the node-centric LG-Flow field is not reconstructable on SBM and Protein, whereas the relational GRAFT field is (Table 5). Thus, our focus is not node-latent transport itself, but constructing a reconstructable node field for the evaluated structurally challenging settings.

Self-conditioning in node-only transport. Self-conditioning feeds a model's own clean-sample estimate back into later predictions and has been used in diffusion and flow models across several domains. Analog Bits (Chen et al., 2022) applies it to discrete diffusion, FlowMol3 (Dunn & Koes, 2026) studies it for molecular Flow Matching, and LG-Flow (Siraudin & Morris, 2026) also uses it in latent graph generation. In GRAFT, we study self-conditioning specifically for relational node-field transport, where graph relations are represented only implicitly by the joint node representation. Our analysis shows that errors in a fixed-width node field can be amplified through relational decoding, while experiments on diverse graph benchmarks show that self-conditioning improves generation, and our MOSES diagnostics show lower relational drift before the terminal state. These observations motivate self-conditioning in GRAFT as a practical mechanism for reducing error accumulation during relational node-field transport.

3 Method

3.1 How Should a Relational Node Field Be Constructed?

The challenge of node-only generation is not removing pair states, but preserving the relational information they carry. We therefore construct a relational node field by modeling relational dependencies before compression and discarding pair states after the field is formed.

We first construct a node field that preserves the relational structure of the input graph. Pairwise and global states are instantiated transiently during encoding, while the resulting representation contains only node latents. A relational decoder is jointly trained to ensure that graph relations remain recoverable after this compression. For a graph with n nodes,

$$G = (V, X, R), \qquad Z = E_\phi(G) \in \mathbb{R}^{n\times d}, \tag{1}$$

where X contains node features and positional encodings, and R contains pairwise relation identities and multi-scale descriptors. The encoder uses pairwise and graph-level reasoning, but retains no pair state afterwards.

Relational encoder. Each candidate pair carries a relation descriptor based on graph diffusion kernels (Kondor & Lafferty, 2002)

$$r_{ij} = \Big[e(\tau_{ij});\ K_{t_1}(i,j), \ldots, K_{t_M}(i,j) \Big], \qquad K_t = \exp(-tL_{\text{norm}}) \,, \tag{2}$$

where $\tau_{ij} \in \{\text{edge}, \text{non-edge}, \text{padding}\}$ and $\{t_m\}_{m=1}^{M}$ are multi-scale diffusion times. Starting from Laplacian positional encodings (Dwivedi & Bresson, 2020), $h_i^{(0)} = F_{\text{in}}(x_i, \text{LPE}_i)$, each refinement step forms transient pair states and folds them back into nodes:

$$\begin{aligned} \pi_{ij}^{(\ell)} &= f_{\text{pair}}\Big(h_i^{(\ell)}, h_j^{(\ell)}, r_{ij}\Big) \,, \qquad m_i^{(\ell)} = \text{Pool}_j\ \pi_{ij}^{(\ell)} \,, \\ g^{(\ell)} &= \text{Pool}_i\ f_{\text{glob}}\Big(h_i^{(\ell)}, m_i^{(\ell)}\Big) \,, \\ h_i^{(\ell+1)} &= h_i^{(\ell)} + \alpha_\ell\, f_{\text{node}}\Big(h_i^{(\ell)}, m_i^{(\ell)}, g^{(\ell)}\Big) \,, \end{aligned} \tag{3}$$

for $\ell = 0, \ldots, L-1$, with parameters shared across steps. After L steps, $Z = F_Z\big(H^{(L)}\big) \in \mathbb{R}^{n \times d}$, and all $\pi_{ij}^{(\ell)}$ and $g^{(\ell)}$ are discarded. Stage I may therefore use $\mathcal{O}(n^2)$ relational computation while its persistent output contains only $\mathcal{O}(nd)$ node variables.

Relational decoder. The decoder mirrors this construction by re-instantiating pair states from Z, which is what makes the node field relational. For each candidate pair,

$$\begin{aligned} q_{ij}^{(0)} &= f_{\text{init}}(z_i + z_j,\ |z_i - z_j|,\ z_i \odot z_j,\ \delta_{ij}) \,, \\ c_i^{(k)} &= \text{Pool}_j\, q_{ij}^{(k)}, \qquad q_{ij}^{(k+1)} = f_{\text{rel}}^{(k)}\Big(q_{ij}^{(k)}, c_i^{(k)}, c_j^{(k)}\Big) \,. \end{aligned} \tag{4}$$

With learned geometric relations δ_{ij}, the refined states predict $\widehat{G} = D_\psi(Z)$. Because these pair states are re-instantiated from Z, rather than inherited from the encoder, graph relations must be recoverable from the joint configuration of the node field.

Field construction objective. We jointly train E_ϕ and D_ψ using

$$\mathcal{L}_{\text{field}} = \mathcal{L}_{\text{relation}} + \mathcal{L}_{\text{aux}}, \tag{5}$$

where $\mathcal{L}_{\text{relation}}$ reconstructs graph relations, while $\mathcal{L}_{\text{aux}}$ supervises domain-specific attributes when present, e.g., atom types in molecular graphs. After Stage I, E_ϕ and D_ψ are frozen, leaving the relational node field as the interface to generation.

3.2 How Should a Relational Node Field Be Transported?

Once constructed, the relational node field must be transported without reintroducing explicit pair states. The challenge is that the transported variables are node-level, but their joint configuration determines a quadratic number of graph relations. Therefore, transport errors in the node field can propagate through relational decoding.

Node-only Flow Matching. After Stage I, we freeze the encoder and decoder and learn Flow Matching (Lipman et al., 2023; Liu et al., 2023) directly over the resulting node field, keeping the entire generative trajectory node-only. For

$$\bar{Z} = \text{Norm}(E_\phi(G)), \qquad X_t = (1-t)X_0 + t\bar{Z}, \quad X_0 \sim \mathcal{N}(0, I), \tag{6}$$

a node-level DiT (Peebles & Xie, 2023) predicts the clean endpoint

$$\widehat{Z}_1 = F_\theta(X_t, t, M), \qquad \mathcal{L}_{\text{transport}} = \mathbb{E}\left[\frac{\|M \odot (\widehat{Z}_1 - \bar{Z})\|_F^2}{N_{\text{valid}} d} \right]. \tag{7}$$

At inference, $\widehat{Z}_1$ defines the endpoint-parameterized flow $v_\theta(X_t, t) = (\widehat{Z}_1 - X_t)/(1-t)$. The final field is denormalized and decoded once by D_ψ. No adjacency, edge, or pair-level state is transported; the persistent generative state size remains $\mathcal{O}(nd)$.

Yet small errors in the transported relational node field can induce much larger relational errors after decoding. We analyze this effect below.

Relational error amplification. Let $S(Z) \in \mathbb{R}^{n\times n}$ denote the continuous relation logits decoded from Z, and define

$$J_D(Z) = \frac{\partial \operatorname{vec} S(Z)}{\partial \operatorname{vec} Z}, \qquad \beta_n(Z) = \frac{\|J_D(Z)\|_F^2}{n(n-1)}, \tag{8}$$

where $\beta_n(Z)$ measures the average squared relational sensitivity.

Proposition 1 (Local relational amplification) *Suppose that S is differentiable at Z and locally Lipschitz in a neighborhood of Z. Let an isotropic perturbation have entries $\Delta Z_{ik} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,\sigma^2)$. Then, for each fixed graph size n,*

$$\lim_{\sigma\to 0} \frac{\mathbb{E}\|S(Z+\Delta Z) - S(Z)\|_F^2}{\mathbb{E}\|\Delta Z\|_F^2} = \frac{n-1}{d}\,\beta_n(Z). \tag{9}$$

If $\beta_n(Z) \geq b > 0$ does not vanish with graph size, the ratio is at least $b(n-1)/d$ and therefore scales as $\Omega(n)$ at fixed latent width.

To see this, let $\delta z = \operatorname{vec}(\Delta Z)$. A first-order expansion gives

$$\operatorname{vec}(S(Z+\Delta Z) - S(Z)) = J_D(Z)\delta z + o(\|\delta z\|). \tag{10}$$

For $\delta z \sim \mathcal{N}(0,\sigma^2 I)$,

$$\mathbb{E}\|J_D\delta z\|_2^2 = \sigma^2\|J_D\|_F^2, \qquad \mathbb{E}\|\delta z\|_2^2 = \sigma^2 nd. \tag{11}$$

Combining these identities proves Eq. equation 9; the complete proof is provided in Appendix A.1. Thus, under the non-degeneracy condition, the aggregate squared relational error is amplified by $\Omega(n)$ at fixed latent width d, or equivalently by $\Omega(\sqrt{n})$ in RMS norm. Importantly, this does not require individual relations to become increasingly unstable: after normalizing by the number of relations and latent coordinates, the explicit size factor disappears and the remaining quantity is precisely $\beta_n(Z)$. The growth therefore arises from a fixed-width node field jointly controlling a quadratic number of relations.

To test this prediction without explicitly forming the full Jacobian, we use a central finite-difference estimate

$$A_n = \frac{\left\|\left[S(Z+\Delta Z) - S(Z-\Delta Z)\right]/2\right\|_F^2}{\|\Delta Z\|_F^2}. \tag{12}$$

For isotropic perturbation directions and sufficiently small perturbations,

$$\mathbb{E}[A_n] \approx \frac{n-1}{d}\,\beta_n(Z). \tag{13}$$

Thus, if the normalized relational sensitivity $\beta_n(Z)$ does not vanish with graph size, A_n should increase with n. Dividing out the explicit size factor gives the normalized estimator

$$\widehat{\beta}_n = \frac{d}{n-1}A_n.$$

Figure 3(a) shows that A_n increases with graph size on held-out Protein graphs. To distinguish genuine relational sensitivity from the explicit number of relations, we remove the $(n-1)/d$ factor and evaluate the normalized sensitivity. As shown in Fig. 3(b), the normalized sensitivity remains substantial over the observed Protein range, providing finite-range empirical support for the assumption that relational sensitivity does not vanish with graph size.

This exposes the central difficulty of node-only transport: reducing the persistent state to $\mathcal{O}(n)$ variables does not remove the relational coupling that the field must preserve.

While Proposition 1 characterizes local perturbations of a relational node field, we further examine whether the same phenomenon appears for errors produced by the learned transport process. Figure 4 shows that actual terminal transport errors exhibit the same size-dependent relational amplification trend, and that the local sensitivity statistic A_n strongly predicts this amplification.

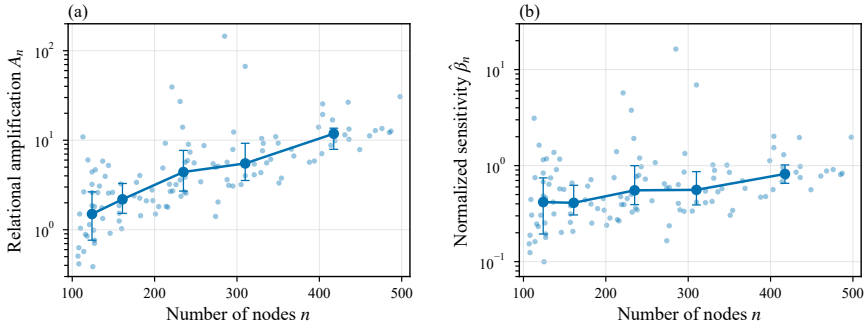


Figure 3: Relational amplification on Protein. (a) A_n increases with graph size. (b) The normalized sensitivity $\widehat{\beta}_n = dA_n/(n-1)$ remains substantial over the observed range, supporting the non-degeneracy assumption in Proposition 1. Light points denote individual graphs; dark markers show size-bin summaries.

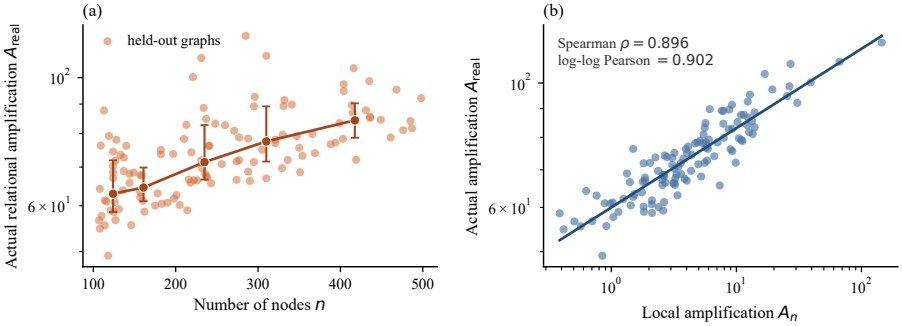


Figure 4: **Actual transport errors exhibit relational amplification.** (a) The amplification of terminal Flow transport errors, measured by A_{real}, increases with graph size on held-out Protein graphs. (b) The local sensitivity statistic A_n from Fig. 3 strongly predicts actual transport amplification.

Self-conditioned transport. Motivated by this sensitivity, we augment the node-only transport with an endpoint-level feedback signal (Chen et al., 2022). With probability p_{sc}, we first obtain a detached endpoint estimate

$$\widetilde{Z}_1 = \text{sg}(F_\theta(X_t, t, M, 0)), \qquad \widehat{Z}_1 = F_\theta(X_t, t, M, \widetilde{Z}_1), \tag{14}$$

and otherwise set the conditioning field to zero. We use $p_{\text{sc}} = 0.5$ throughout. During sampling, the previous endpoint estimate is carried forward as the conditioning field for the next solver step.

Self-conditioning changes the transport dynamics by feeding the previous endpoint estimate back into the next prediction. While this feedback does not guarantee contraction, it provides a state-dependent feedback signal for reducing error accumulation during node-field transport. A complete analysis of this feedback in terms of block Jacobians is provided in Appendix A.4.

Self-conditioning introduces no persistent pair state: the generative trajectory remains node-only. Proposition 1 explains why perturbations of a relational node field can have increasingly large aggregate effects after decoding; we evaluate the effect of self-conditioning on transport error and generation quality empirically in Sec. 4.3.2.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate six benchmarks spanning synthetic, large-scale, and molecular graphs. The synthetic benchmarks are Extended Planar and Extended Tree (Krimmel et al., 2025), each with 64 nodes, and the canonical SBM (Martinkus et al., 2022) split of 128 training, 32 validation, and 40 test graphs. Protein (Bergmeister et al., 2024) is the large-graph benchmark; its 184 test graphs average

Table 1: Generation quality on molecular graph benchmarks.

	GuacaMol					MOSES						
Model	V.↑	V.U.↑	V.U.N.↑	KL↑	FCD↑	Val.↑	Uniq.↑	Nov.↑	Filt.↑	FCD↓	SNN↑	Scaf.↑
Train	100.0	100.0	0.0	99.9	92.8	100.0	100.0	0.0	100.0	0.01	0.64	99.1
DiGress	85.2	85.2	85.1	92.9	68.0	85.7	**100.0**	95.0	97.1	1.19	0.52	14.8
DisCo	86.6	86.6	86.5	92.6	59.7	88.3	**100.0**	**97.7**	95.6	1.44	0.50	15.1
Cometh	98.9	98.9	97.6	96.7	72.7	90.5	99.9	92.6	**99.1**	1.27	0.54	**16.0**
DeFoG	**99.0**	**99.0**	**97.9**	**97.7**	73.8	92.8	99.9	92.1	98.9	1.95	0.55	14.4
LG-Flow	93.0	93.0	91.5	97.2	81.8	88.4	99.9	90.5	98.9	1.43	0.55	12.3
Ours	93.5	93.5	92.3	**97.7**	**84.0**	**94.4**	**100.0**	92.7	98.8	**0.80**	**0.57**	12.4

Table 2: Generation quality on generic graph benchmarks.

Dataset	Extended Planar		Extended Tree		SBM	
Method	V.U.N. (↑)	Ratio (↓)	V.U.N. (↑)	Ratio (↓)	V.U.N. (↑)	Ratio (↓)
Training Set	100.0	1.0	100.0	1.0	85.9	1.0
DiGress	–	–	–	–	60.0	**1.7**
Cometh	96.7±0.8	5.1	93.8±1.6	2.0	75.0±3.7	–
DeFoG	**99.6±0.4**	2.7	98.1±0.7	**1.7**	**90.0±5.1**	4.9±1.3
SparseDiff	74.1±2.1	5.8	95.3±1.5	**1.7**	56.0±8.5	–
LG-Flow†	99.4±0.3	2.7	**98.3±0.6**	2.3	–	–
Ours	98.7±1.0	**2.5**	96.3±1.7	2.1	77.0±3.7	2.1±0.5

† LG-Flow could not be trained on SBM because its reconstruction rate is constantly 0.

250.73 nodes, with a maximum of 490 nodes. The molecular benchmarks are MOSES (Polykovskiy et al., 2020) and GuacaMol (Brown et al., 2019). We use the standard data split for each benchmark. Protein also supports the scalability and sampling-efficiency experiments because its graphs are substantially larger than those in the other benchmarks.

Evaluation metrics. Following DeFoG (Qin et al., 2025), we report Valid/Unique/Novel (V.U.N.) and the normalized MMD Ratio on Extended Planar, Extended Tree, and SBM. Following AutoGraph (Chen et al., 2025), we additionally report Degree-, Clustering-, Orbit-, and Spectral-MMD together with the Ratio on Protein. For molecular graphs, we follow the standard MOSES (Polykovskiy et al., 2020) and GuacaMol (Brown et al., 2019) metrics, including validity, uniqueness, novelty, FCD, SNN, Frag, Scaf, and KL where applicable.

Baselines. We compare against discrete diffusion methods (DiGress (Vignac et al., 2023), DisCo (Xu et al., 2024), Cometh (Siraudin et al., 2024), and SparseDiff (Qin et al., 2023)) and the discrete flow-matching model DeFoG (Qin et al., 2025). LG-Flow (Siraudin & Morris, 2026) is our direct node-latent baseline: it keeps a node-only persistent state but constructs it through near-lossless adjacency reconstruction. For Protein, we additionally compare with the autoregressive AutoGraph (Chen et al., 2025), the spectral generator SPECTRE (Martinkus et al., 2022), and ESGG (Bergmeister et al., 2024). Best results are bold; second-best results are underlined throughout the result tables.

4.2 Generation Quality

Generation quality. GRAFT achieves the best FCD among the compared methods on both GuacaMol and MOSES (Table 1), while remaining competitive on Extended Planar and Tree (Table 2). On SBM and Protein, the direct node-latent baseline LG-Flow does not obtain a reconstructable Stage-I representation under our protocol, whereas GRAFT provides a reconstructable node-only result (Tables 2–3). On Protein, GRAFT achieves a Ratio of 5.1 while retaining the lower sampling cost analyzed in Sec. 4.4.

4.3 Analysis

4.3.1 Relational Field Construction

Relational field construction. Table 5 isolates the role of relational construction. Panel (a) compares LG-Flow and GRAFT under the matched 200K-step Stage-I training protocol; panel (b) ablates relational field construction within GRAFT.7

Table 3: Generation quality on Protein.

Method	**Deg.↓**	**Clus.↓**	**Orbit↓**	**Spec.↓**	**Ratio↓**
Training Set	0.0003	0.0068	0.0032	0.0005	1.0
SPECTRE	0.0056	0.0843	0.0267	0.0052	12.5
SparseDiff	0.0031	**0.0228**	0.0482	0.0014	7.6
DiGress	0.0041	0.0489	0.1286	0.0018	16.2
ESGG	0.0030	0.0309	**0.0047**	**0.0013**	4.7
AutoGraph	**0.0004**	0.0244	0.0056	**0.0013**	**2.3**
LG-Flow†	–	–	–	–	–
Ours	0.0010	0.0514	0.0209	0.0025	5.1

† LG-Flow could not be trained on Protein because its reconstruction rate is constantly 0.

Table 4: Protein quality and efficiency.

Method	**Ratio↓**	**Time/graph (s)↓**	**Peak Mem. (MiB)↓**
DiGress	16.2	43.137±21.117	630.8
AutoGraph	**2.3**	6.295±1.970	544.3
LG-Flow†	–	0.599±0.108	211.4
Ours	5.1	**0.290±0.027**	**110.6**

All methods use their native inference configuration, with batch size 1 on an RTX 4090. † No valid Protein quality result due to failed Stage-I reconstruction.

Table 5: Stage-I exact reconstruction rates.

(a) Matched construction comparison

Dataset	LG-Flow	GRAFT
Extended Planar	99.61	**100.00**
Extended Tree	98.44	**100.00**
MOSES	99.67	**100.00**
GuacaMol	99.73	**100.00**
SBM	0.00	**96.88**
Protein	0.00	**93.75**

(b) Relation ablation

Dataset	w/o Rel.	w/ Rel.
Extended Planar	**100.00**	**100.00**
Extended Tree	98.05	**100.00**
MOSES	99.75	**100.00**
GuacaMol	99.42	**100.00**
SBM	0.00	**96.88**
Protein	0.00	**93.75**

Exact reconstruction rate is reported in percent, and higher is better. On Extended Planar, Tree, MOSES, and GuacaMol, both LG-Flow and relation-free GRAFT already achieve near-lossless reconstruction. On SBM and Protein, however, both collapse to 0% exact reconstruction, while relational field construction restores 96.88% and 93.75%, respectively, showing that the gain comes from how relational information is folded into the node field rather than from decoder capacity alone.

4.3.2 Self-Conditioned Transport

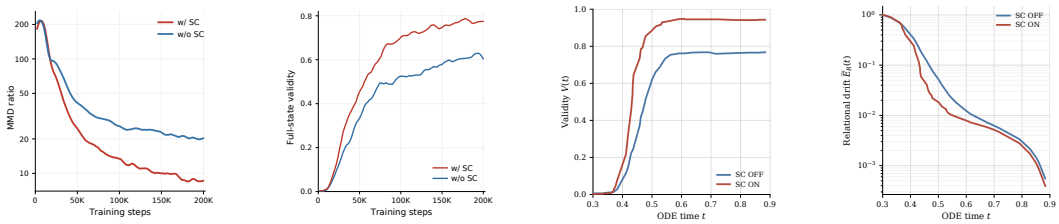
We separate the training-time and inference-time effects of self-conditioning (SC). Figure 5(a,b) compares two *independently trained* models under matched training budgets: the *w/ SC* model is trained with $p_{\mathrm{sc}} = 0.5$ and sampled with one-step-lagged endpoint feedback, whereas the *w/o SC* model is trained with $p_{\mathrm{sc}} = 0$ and sampled with a zero conditioning field throughout. Thus, Fig. 5(a,b) measures the end-to-end benefit of self-conditioning rather than toggling an input of a fixed checkpoint. Across the two training runs, self-conditioning improves the primary metric on all six datasets.

Reducing amplified transport error. Proposition 1 implies that node-field error should be controlled before relational decoding. Self-conditioning feeds each endpoint estimate into subsequent predictions to reduce error accumulation. On MOSES, Figure 5(c,d) shows earlier decoder-valid endpoints and lower normalized relational drift to the terminal state through most of the transition interval. Thus, less transport error reaches the decoder and is subsequently amplified.

Table 6 isolates inference-time feedback using the same SC-trained checkpoint. Gains exceed 20% across datasets and reach 46.7% on SBM, while the Protein Ratio falls by 65%, from 14.75 to 5.10. Together, these results show that self-conditioning reduces amplified error while keeping transport node-only.

4.4 Sampling Efficiency

Sampling scalability. Across the graph-size ranges evaluated in Fig. 1, GRAFT exhibits slower growth than the representative pair-state generators in Protein sampling time and peak memory under controlled matched-100-NFE inference, consistent with transporting $\mathcal{O}(nd)$ node states rather than repeatedly updating $\mathcal{O}(n^2)$ pair states.



(a) Protein training (b) MOSES training (c) MOSES validity (d) MOSES relational drift

Figure 5: **Self-conditioning (SC) improves matched training runs and reduces transport error.** (a,b) w/ SC and w/o SC denote independently trained models with and without self-conditioning, respectively, under matched architecture, budget, and evaluation protocol. (c,d) For the same SC-trained checkpoint over 512 MOSES samples, SC ON/OFF toggles feedback only during sampling; SC yields earlier decoder-valid endpoints and lower normalized relational drift to the terminal state through most of the transition interval.

Table 6: Inference-time self-conditioning (SC) ablation.

Dataset	Metric	w/o SC	w/ SC
Extended Planar	V.U.N. ↑	$\underline{79.30 \pm 4.90}$	$\mathbf{98.70 \pm 1.00}$
Extended Tree	V.U.N. ↑	$\underline{80.08 \pm 4.28}$	$\mathbf{96.30 \pm 1.70}$
SBM	V.U.N. ↑	$\underline{52.50 \pm 6.12}$	$\mathbf{77.00 \pm 3.70}$
Protein	Ratio ↓	$\underline{14.75 \pm 1.58}$	$\mathbf{5.10 \pm 0.46}$
MOSES	Val. ↑	$\underline{76.16 \pm 0.21}$	$\mathbf{94.38 \pm 0.07}$
GuacaMol	V. ↑	$\underline{77.33 \pm 0.45}$	$\mathbf{93.51 \pm 0.38}$

Single-graph efficiency. Under native Protein inference configurations, GRAFT requires only 0.290 ± 0.027 s per graph and 110.6 MiB of peak GPU memory, compared with 0.599 s / 211.4 MiB for LG-Flow, 6.295 s / 544.3 MiB for AutoGraph, and 43.137 s / 630.8 MiB for DiGress (Table 4). GRAFT therefore provides the lowest latency and memory footprint among the evaluated methods. The per-method sampling protocol is listed in Table 13.

5 Conclusion

We introduced GRAFT, a graph generation framework that separates relational reasoning from generative transport. Rather than carrying adjacency, edge, or pair-level states throughout sampling, GRAFT uses transient pairwise and global reasoning to construct a relational node field and then performs Flow Matching entirely in the resulting node space (Lipman et al., 2023; Liu et al., 2023). Our experiments show that reconstruction-oriented node representations are sufficient on simpler benchmarks, but relational field construction is decisive on SBM and Protein: the node-centric and relation-free controls in our matched analysis do not remain reconstructable there. We further analyze how errors in a fixed-width node field can be amplified through relational decoding and observe the same amplification trend for actual Flow transport errors on Protein. We evaluate self-conditioning (Chen et al., 2022; Dunn & Koes, 2026) separately as a practical mechanism for reducing transport error before relational decoding. Across synthetic, molecular, and large-graph benchmarks, GRAFT achieves competitive generation quality while retaining a fully node-only generative trajectory and reducing sampling time and memory on Protein.

While GRAFT eliminates pair-level states throughout generative transport, pairwise computation remains in relational field construction and final graph realization. Scaling these stages to extremely large graphs is an important direction for future work.

AI use statement

Generative AI tools were used for language editing and formatting of tables, figures, and LaTeX. All methods, theoretical analyses, and experimental results were developed and produced by the authors, who take full responsibility for the content of this paper.

REFERENCES

Andreas Bergmeister, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Efficient and scalable graph generation through iterative local expansion. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2312.11529.

Nathan Brown, Marco Fiscato, Marwin H. S. Segler, and Alain C. Vaucher. Guacamol: Benchmarking models for de novo molecular design. *Journal of Chemical Information and Modeling*, 59(3):1096–1108, 2019. doi: 10.1021/acs.jcim.8b00839.

Dexiong Chen, Markus Krimmel, and Karsten Borgwardt. Flatten graphs as sequences: Transformers are scalable graph generators. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. *arXiv preprint arXiv:2208.04202*, 2022.

Xiaohui Chen, Jiaxing He, Xu Han, and Li-Ping Liu. Efficient and degree-guided graph generation via discrete diffusion modeling. *arXiv preprint arXiv:2305.04111*, 2023.

Ian Dunn and David R Koes. Flowmol3: flow matching for 3d de novo small-molecule generation. *Digital Discovery*, 5(5):2052–2066, 2026.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020. URL https://arxiv.org/abs/2012.09699.

Xingcheng Fu, Yisen Gao, Yuecen Wei, Qingyun Sun, Hao Peng, Jianxin Li, and Xianxian Li. Hyperbolic geometric latent diffusion model for graph generation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 14102–14124, 2024. URL https://proceedings.mlr.press/v235/fu24c.html.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 10362–10383. PMLR, 2022. URL https://proceedings.mlr.press/v162/jo22a.html.

Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the 19th International Conference on Machine Learning*, pp. 315–322. Morgan Kaufmann, 2002.

Markus Krimmel, Jenna Wiens, Karsten M. Borgwardt, and Dexiong Chen. Towards fast graph generation via autoregressive noisy filtration modeling. *arXiv preprint arXiv:2502.02415*, 2025.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.02747.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2209.03003.

Karolis Martinkus, Andreas Loukas, Nathanaël Perraudin, and Roger Wattenhofer. Spectre: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *International Conference on Machine Learning*, pp. 15159–15179. PMLR, 2022.

Giorgia Minello, Alessandro Bicciato, Luca Rossi, Andrea Torsello, and Luca Cosmo. Generating Graphs via Spectral Diffusion. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2402.18974.

Van Khoa Nguyen, Yoann Boget, Frantzeska Lavda, and Alexandros Kalousis. GLAD: Improving latent graph generative modeling with simple quantization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 19695–19702, 2025. doi: 10.1609/aaai.v39i18.34169. URL https://ojs.aaai.org/index.php/AAAI/article/view/34169.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, Artur Kadurin, Simon Johansson, Hongming Chen, Sergey Nikolenko, Alán Aspuru-Guzik, and Alex Zhavoronkov. Molecular sets (MOSES): A benchmarking platform for molecular generation models. *Frontiers in Pharmacology*, 11:565644, 2020. doi: 10.3389/fphar.2020.565644.

Yiming Qin, Clément Vignac, and Pascal Frossard. Sparse training of discrete diffusion models for graph generation. *arXiv preprint arXiv:2311.02142*, 2023.

Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. Defog: Discrete flow matching for graph generation. In *International Conference on Machine Learning (ICML)*, 2025.

Antoine Siraudin and Christopher Morris. Principled latent diffusion for graphs via laplacian autoencoders. *arXiv preprint arXiv:2601.13780*, 2026.

Antoine Siraudin, Fragkiskos D. Malliaros, and Christopher Morris. Cometh: A continuous-time discrete-state graph diffusion model. *arXiv preprint arXiv:2406.06449*, 2024.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. In *International Conference on Learning Representations (ICLR)*, 2023.

Zhe Xu, Ruizhong Qiu, Yuzhong Chen, Huiyuan Chen, Xiran Fan, Menghai Pan, Zhichen Zeng, Mahashweta Das, and Hanghang Tong. Discrete-state continuous-time diffusion for graph generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Cai Zhou, Xiyuan Wang, and Muhan Zhang. Unifying Generation and Prediction on Graphs with Latent Graph Diffusion. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/718d02a76d69686a36eccc8cde3e6a41-Abstract-Conference.html.

A Relational Sensitivity and Transport Analysis

This appendix provides the detailed analysis underlying the relational sensitivity discussion in the main text. We first characterize how a small perturbation of a node field propagates through a relational decoder, then establish the correspondence between this sensitivity and the finite-difference estimator used in our experiments. Finally, we analyze error propagation along endpoint-based transport and describe how self-conditioning modifies the local feedback dynamics.

A.1 Local Relational Error Amplification

Let

$$Z \in \mathbb{R}^{n \times d}$$

denote a standardized node field, and let

$$S(Z) \in \mathbb{R}^{n \times n}$$

denote the continuous relation logits produced by the frozen Stage-I decoder. The mapping S includes the fixed denormalization from standardized transport coordinates to the latent coordinates expected by the decoder. We set the diagonal entries of S to zero and consider all ordered off-diagonal pairs.

Although only node states are transported, the decoded relations are not node-independent. In particular, the contextual decoder can make each relation score depend on the entire node field,

$$S_{ij} = d_\psi(z_i, z_j; Z).$$

We define the average squared relational sensitivity as

$$\beta_n(Z) = \frac{1}{n(n-1)} \sum_{i \neq j} \|\nabla_Z S_{ij}(Z)\|_F^2 . \tag{15}$$

Equivalently, if

$$J_D(Z) = \frac{\partial \operatorname{vec} S(Z)}{\partial \operatorname{vec} Z},$$

then

$$\beta_n(Z) = \frac{\|J_D(Z)\|_F^2}{n(n-1)}. \tag{16}$$

Non-degeneracy assumption. We assume that, over the family of node fields under consideration, there exist constants $b > 0$ and $n_0 \geq 2$, independent of graph size, such that

$$\beta_n(Z) \geq b, \qquad n \geq n_0. \tag{17}$$

This assumption states that the average sensitivity of a decoded relation does not vanish as graph size increases.

Proof of Proposition 1. Let

$$\xi = \mathrm{vec}(\Delta Z)/\sigma,$$

so that

$$\xi \sim \mathcal{N}(0, I_{nd}).$$

Differentiability of S at Z gives

$$\mathrm{vec}(S(Z + \Delta Z) - S(Z)) = \sigma J_D(Z)\xi + r_\sigma, \tag{18}$$

where

$$\mathbb{E}\|r_\sigma\|_2^2 = o(\sigma^2).$$

The latter follows from the local differentiability and Lipschitz assumptions; Gaussian tails outside any fixed local neighborhood vanish faster than σ^2 as $\sigma \to 0$.

Since

$$\mathbb{E}[\xi\xi^\top] = I,$$

we have

$$\mathbb{E}\|\sigma J_D \xi\|_2^2 = \sigma^2 \|J_D\|_F^2. \tag{19}$$

The cross term between $\sigma J_D \xi$ and r_σ is $o(\sigma^2)$ by the Cauchy–Schwarz inequality. Hence

$$\mathbb{E}\|S(Z + \Delta Z) - S(Z)\|_F^2 = \sigma^2 \|J_D\|_F^2 + o(\sigma^2). \tag{20}$$

Meanwhile,

$$\mathbb{E}\|\Delta Z\|_F^2 = \sigma^2 nd. \tag{21}$$

Therefore,

$$\lim_{\sigma \to 0} \frac{\mathbb{E}\|S(Z + \Delta Z) - S(Z)\|_F^2}{\mathbb{E}\|\Delta Z\|_F^2} = \frac{\|J_D\|_F^2}{nd} \tag{22}$$

$$= \frac{n-1}{d}\beta_n(Z), \tag{23}$$

which proves Eq. equation 9. Applying Eq. equation 17 gives the claimed lower bound. □

Interpretation. The result concerns aggregate relational error rather than the average error of a single relation. Indeed, dividing the output error by the number of ordered relations $n(n-1)$ and the input error by the number of latent coordinates nd removes the explicit factor of n and leaves precisely the normalized sensitivity $\beta_n(Z)$. Thus, the graph-size factor arises from the number of relations influenced by a fixed-width node field, not from an assumption that each individual relation becomes increasingly unstable.

The limit above is local and is taken for each fixed n. Extending the claim to a finite perturbation radius uniformly over graph size would additionally require uniform control of the higher-order remainder.

A.2 Finite-Difference Estimation of Relational Sensitivity

The empirical analysis in the main paper uses central finite differences to estimate the normalized sensitivity in Eq. equation 15 without explicitly constructing the full decoder Jacobian.

For a perturbation ΔZ, define

$$A_n^{\mathrm{cen}} = \frac{\left\| [S(Z + \Delta Z) - S(Z - \Delta Z)]/2 \right\|_F^2}{\|\Delta Z\|_F^2}. \tag{24}$$

We then define the normalized estimator

$$\widehat{\beta}_n = \frac{d}{n-1} A_n^{\mathrm{cen}}. \tag{25}$$

To show why this quantity estimates $\beta_n(Z)$, let

$$\mathrm{vec}(\Delta Z) = \sigma\sqrt{nd}\, u, \tag{26}$$

where u is uniformly distributed on the unit sphere in $\mathbb{R}^{nd}$. Then

$$\mathbb{E}[uu^\top] = \frac{1}{nd} I. \tag{27}$$

For sufficiently small σ, the central difference cancels the leading even-order terms,

$$\frac{\mathrm{vec}\, S(Z+\Delta Z) - \mathrm{vec}\, S(Z-\Delta Z)}{2} = J_D(Z)\,\mathrm{vec}(\Delta Z) + o(\|\Delta Z\|). \tag{28}$$

Consequently,

$$\lim_{\sigma\to 0} \mathbb{E}_u A_n^{\mathrm{cen}} = \mathbb{E}_u \|J_D u\|_2^2 \tag{29}$$

$$= \frac{\|J_D\|_F^2}{nd}. \tag{30}$$

Combining this result with Eq. equation 25 gives

$$\lim_{\sigma\to 0} \mathbb{E}_u \widehat{\beta}_n = \frac{\|J_D\|_F^2}{n(n-1)} = \beta_n(Z). \tag{31}$$

Thus, the normalized finite-difference quantity plotted in the main experiments estimates exactly the sensitivity factor appearing in Proposition 1. The empirical results are used only to examine whether this factor remains non-vanishing over the observed graph-size range; they do not constitute a proof of a uniform lower bound.

In Fig. 3(b), we report this normalized estimator on 121 held-out Protein graphs. It remains non-vanishing over the observed graph-size range; this provides empirical support for the non-degeneracy assumption over the evaluated range, rather than a proof of a uniform asymptotic lower bound.

A.3 Transport-Error Propagation

We next separate decoder sensitivity from error accumulation during Stage-II transport. Let

$$x_k \in \mathbb{R}^{nd}$$

denote the vectorized standardized node field at integration step k, and let

$$c_k \in \mathbb{R}^{nd}$$

denote the self-conditioning state.

The endpoint-parameterized sampler can be written in the generic form

$$f_k = F_\theta(x_k, t_k, c_k), \qquad x_{k+1} = (1-a_k)x_k + a_k f_k, \tag{32}$$

where a_k is determined by the numerical integration schedule. The exact time grid and denominator clipping used in our implementation are reported in Appendix B.

Let

$$y_k = (x_k, c_k)$$

denote the complete sampling state, and write the corresponding one-step mapping as

$$y_{k+1} = T_k(y_k).$$

Consider an arbitrary reference trajectory

$$y_0^*, y_1^*, \ldots, y_K^*.$$

Suppose that T_k is locally λ_k-Lipschitz over the region containing the compared trajectories and that the reference trajectory has one-step residual

$$\|T_k(y_k^*) - y_{k+1}^*\| \le \varepsilon_k. \tag{33}$$

Then

$$\|y_{k+1} - y_{k+1}^*\| \le \lambda_k \|y_k - y_k^*\| + \varepsilon_k. \tag{34}$$

Repeated application yields

$$\|y_K - y_K^*\| \le \left(\prod_{\ell=0}^{K-1} \lambda_\ell \right) \|y_0 - y_0^*\| + \sum_{j=0}^{K-1} \varepsilon_j \prod_{\ell=j+1}^{K-1} \lambda_\ell. \tag{35}$$

This expression is an upper bound on propagated error; it does not imply that sampling errors necessarily grow. Error growth or contraction depends on the learned local dynamics represented by the factors λ_k.

Let $x_\star$ denote a target endpoint field and let

$$\overline{S}(x) = S(\mathrm{unvec}(x))$$

denote the complete standardized-coordinate decoder. If $\overline{S}$ is $L_{\overline{S}}$-Lipschitz on the relevant segment, then

$$\|\overline{S}(x_K) - \overline{S}(x_\star)\|_F \le L_{\overline{S}} \|x_K - x_\star\|. \tag{36}$$

For small random terminal errors

$$e_K = x_K - x_\star$$

with covariance

$$C_K = \mathbb{E}[e_K e_K^\top],$$

the linearized expected relational error is more precisely

$$\mathbb{E}\|\overline{S}(x_K) - \overline{S}(x_\star)\|_F^2 \approx \mathrm{tr}\left(\overline{J}_D C_K \overline{J}_D^\top \right), \tag{37}$$

where $\overline{J}_D$ is the decoder Jacobian with respect to standardized transport coordinates. Proposition 1 corresponds to the isotropic special case of this expression.

Equations equation 35 and equation 37 therefore separate two effects: the Stage-II transport determines the distribution of terminal field error, while the frozen relational decoder determines how that field error is converted into relational error.

A.4 Self-Conditioning as Feedback

Self-conditioning modifies the transport dynamics by augmenting the state with an endpoint estimate. At inference time, let

$$f_k = F_\theta(x_k, t_k, c_k). \tag{38}$$

With self-conditioning, the predicted endpoint is propagated to the next integration step,

$$c_{k+1} = f_k, \tag{39}$$

whereas without self-conditioning the conditioning input is fixed to zero.

To characterize the local effect of this feedback, define

$$J_x = \frac{\partial F_\theta}{\partial x}, \qquad J_c = \frac{\partial F_\theta}{\partial c}. \tag{40}$$

Linearizing the augmented update around a reference state gives the block Jacobian

$$B_k = \begin{bmatrix} (1 - a_k)I + a_k J_x & a_k J_c \\ J_x & J_c \end{bmatrix}. \tag{41}$$

Therefore, self-conditioning changes the ordered product of local Jacobians governing perturbation propagation through the sampler. A sufficient local contraction condition is

$$\|B_k\| < 1 \tag{42}$$

throughout the relevant neighborhood, but this property does not follow from self-conditioning alone. More generally, whether feedback reduces terminal sensitivity depends on the learned matrices J_x, J_c, and their ordered products along the generated trajectory.

Training-time and sampling-time self-conditioning should also be distinguished. During training, the conditioning signal is obtained through a detached preliminary prediction at the same noisy state. During sampling, the conditioning state is the endpoint prediction inherited from the previous integration step. The exact implementation is given in Appendix B.

Accordingly, the analysis does not claim that self-conditioning universally stabilizes transport. Rather, it identifies self-conditioning as a feedback mechanism that changes the temporal error-propagation operator. Whether this change improves relational-field generation is an empirical property of the trained model and is evaluated in the main experiments.

A.5 Permutation Equivariance of Relational Field Construction

Let G be a graph with Laplacian L, and let P be an arbitrary permutation matrix. Under node relabeling,

$$L' = PLP^\top. \tag{43}$$

For a simple eigenvalue λ_k, the corresponding eigenvector of L' is defined up to a global sign,

$$u'_k = s_k P u_k, \qquad s_k \in \{-1, +1\}. \tag{44}$$

Directly using u_k as a node positional encoding therefore introduces a spectral sign gauge: independently recomputing the eigendecomposition after permutation may change the latent representation even when the underlying graph is unchanged up to relabeling.

To remove this ambiguity, GRAFT uses sign-invariant spectral descriptors. For the node branch, each eigenvector is represented through its elementwise magnitude,

$$\phi_{ik}^{\text{node}} = |u_{ik}|, \tag{45}$$

which transforms as

$$|u'_k| = |s_k P u_k| = P|u_k|. \tag{46}$$

For pairwise spectral relations, we use the mode-wise projector entries

$$\phi_{ij,k}^{\text{pair}} = u_{ik} u_{jk}. \tag{47}$$

These features are invariant to the global sign of each eigenvector and equivariant to node relabeling:

$$u'_{ik} u'_{jk} = (Pu_k)_i (Pu_k)_j, \tag{48}$$

or, in matrix form,

$$u'_k u'^\top_k = P u_k u_k^\top P^\top. \tag{49}$$

The remaining heat-kernel relations satisfy the same transformation law. Since

$$K_t = \exp(-tL), \tag{50}$$

we have

$$K'_t = \exp(-tPLP^\top) = PK_tP^\top. \tag{51}$$

The subsequent relational encoder consists only of shared pair updates, permutation-invariant aggregation over neighboring pairs, shared node updates, and graph-level invariant pooling. Therefore, conditioned on these sign-invariant spectral inputs, the resulting node field satisfies

$$E(PGP^\top) = PE(G). \tag{52}$$

Likewise, the contextual relational decoder is permutation equivariant because pair states are constructed through shared symmetric functions of (z_i, z_j), followed by permutation-invariant pair-to-node aggregation. Hence

$$D(PZ) = PD(Z)P^\top. \tag{53}$$

Consequently, the complete Stage-I mapping is permutation equivariant under node relabeling, provided the spectral modes are simple.

Table 7: Shared Stage-I encoder hyperparameters.

Parameter	Value
Node hidden dimension	256
Pair hidden dimension	96
Global hidden dimension	256
Relational refinement iterations	4
Parameter sharing across iterations	Yes
Pair pooling seeds	4
Pair pooling output dimension	64
Latent dimension	32
Latent type	Deterministic
Persistent graph latent	None
Initial normalization	LayerNorm
Node residual gate	Sigmoid gate
Dropout	0.0

Empirical gauge audit. To identify the source of permutation inconsistency in the original raw-LPE implementation, we independently recomputed the eigendecomposition after random node permutations. The raw representation yielded a mean encoder discrepancy of $r_{\text{enc}} = 0.786$. Oracle alignment of eigenvector signs reduced this discrepancy to 1.44×10^{-5}. Block-wise orthogonal Procrustes alignment produced the same value, and all inspected eigenspaces were one-dimensional under a relative eigenvalue tolerance of 10^{-4}. These results indicate that the observed inconsistency is almost entirely due to the $\pm$ sign gauge rather than the relational encoder or rotations within degenerate eigenspaces.

For repeated eigenvalues, individual eigenvectors are not uniquely defined up to sign but up to an orthogonal transformation within the eigenspace. In that case, the corresponding block projector $U_B U_B^\top$ provides the gauge-invariant generalization.

B Implementation Details

B.1 Stage-I Encoder Configuration

All datasets use the same Stage-I relational-field encoder backbone. The encoder consists of an LPE node branch followed by a relational pair-to-node encoder. Pairwise heat-kernel features and transient global states are used internally during field construction, while only the final node field is retained.

The relational encoder uses node, pair, and global hidden dimensions of 256, 96, and 256, respectively. It applies four parameter-shared refinement iterations, with a learned step-specific scaling coefficient at each iteration. The resulting node representation is mapped to a deterministic 32-dimensional node latent. The global representation is used only during encoding and is discarded afterwards; therefore, the persistent Stage-I output is a node field

$$Z \in \mathbb{R}^{N \times 32}.$$

No graph-level latent, variational posterior, or KL regularization is used.

The node branch uses Laplacian positional encoding (LPE), while pairwise structural information is provided through heat-kernel features. For LPE, we use the unnormalized graph Laplacian $L = D - A$ and normalize the eigenvectors before encoding. Heat-kernel pair features are instead computed from the normalized Laplacian

$$L_{\text{norm}} = I - D^{-1/2} A D^{-1/2},$$

using eight diffusion scales

$$\{0.125,\ 0.25,\ 0.5,\ 1,\ 2,\ 4,\ 8,\ 16\}.$$

No RRWP features are used.

The dataset-specific encoder parameters are summarized in Table 8. These differences only affect the input spectral representation; the relational refinement backbone, latent width, and number of

Table 8: Dataset-specific Stage-I encoder parameters.

Dataset	LPE dim	K_{LPE}	Modulation	Atom types
Planar / Tree	256	16	0.005	–
SBM	256	64	0.005	–
Protein	256	64	0.000	–
MOSES	384	16	0.050	7
GuacaMol	384	32	0.050	12

refinement iterations remain unchanged. LPE dim denotes the output dimension of the spectral node encoder, and K_{LPE} denotes the number of Laplacian eigenvectors used by the LPE branch.

For non-molecular datasets, the spectral node representation is constructed without categorical node-type embeddings. For MOSES and GuacaMol, the same encoder architecture is retained, while the atom-type vocabulary and spectral width are adjusted to the corresponding dataset. MOSES uses 7 atom types and 16 Laplacian eigenvectors, whereas GuacaMol uses 12 atom types and 32 eigenvectors.

Within each refinement iteration, transient pair states are formed from a three-way pair-type embedding (non-edge, edge, and padding) together with the eight heat-kernel channels. Pairwise information is pooled back to node states, and a transient global state further provides graph-level context. Both pair and global states are internal to the encoder and are discarded after the final refinement step.

B.2 Stage-I Decoder Configuration

All datasets use the same contextual relation decoder backbone. The decoder receives only the node field

$$Z = \{z_i\}_{i=1}^{N}, \qquad z_i \in \mathbb{R}^{32},$$

and reconstructs graph relations by re-instantiating pairwise states from the joint node representation. No pair-level state produced by the encoder is passed to the decoder.

Pair initialization. Each node latent is first mapped to a learned three-dimensional coordinate through

$$c_i = f_{\mathrm{coord}}(z_i),$$

where the coordinate network is a two-layer MLP,

$$32 \to 64 \to 3,$$

with a SiLU activation between the two linear layers.

For each candidate pair (i, j), we construct a symmetric pair feature

$$u_{ij} = [z_i + z_j,\ |z_i - z_j|,\ z_i \odot z_j,\ d_{ij}],$$

where

$$d_{ij} = \|c_i - c_j\|_2.$$

Since each node latent has dimension 32, the resulting pair feature has dimension

$$3 \times 32 + 1 = 97.$$

It is projected to the decoder pair space through

$$97 \to 192 \to 128,$$

again using SiLU activation. The resulting initial pair state is $p_{ij}^{(0)} \in \mathbb{R}^{128}$ and is explicitly symmetrized with respect to (i, j).

Contextual pair refinement. The initial pair state is refined by three `ContextualPairBlock`s. Each block maintains a 128-dimensional pair state and uses four context heads, each with a value dimension of 32.

Table 9: Shared Stage-I decoder architecture.

Parameter	Configuration
Input node latent dimension	32
Coordinate MLP	$32 \to 64 \to 3$
Coordinate dimension	3
Pair initialization features	$z_i + z_j,\ \lvert z_i - z_j \rvert,\ z_i \odot z_j,\ d_{ij}$
Pair initialization dimension	97
Pair initialization MLP	$97 \to 192 \to 128$
Pair hidden dimension	128
Contextual refinement blocks	3
Context heads per block	4
Value dimension per head	32
Node-context dimension	128
Score projection	$128 \to 4$
Value projection	$128 \to 128$
Pair-update input dimension	512
Pair-update MLP	$512 \to 256 \to 128$
Residual scale	0.05
Normalization	LayerNorm(128)
Activation	SiLU
Dropout	0.0
Pair symmetrization	Yes

For a pair state p_{ij}, the block predicts four scalar context scores and four 32-dimensional value vectors,

$$\text{score} : 128 \to 4, \qquad \text{value} : 128 \to 4 \times 32.$$

After masked aggregation over neighboring candidate pairs, the four heads are concatenated to form a 128-dimensional node context m_i.

The pair update is then constructed from

$$[p_{ij},\ m_i + m_j,\ |m_i - m_j|,\ m_i \odot m_j]\,,$$

which has total dimension

$$128 \times 4 = 512.$$

The contextual pair update is produced by

$$512 \to 256 \to 128.$$

Each refinement block uses a residual update with scale 0.05, followed by LayerNorm and pair symmetrization. No dropout is used in the decoder.

Relation prediction. For non-molecular datasets, the final pair state is mapped to a single edge margin using

$$f_{\text{edge}} : \mathbb{R}^{128} \to \mathbb{R}.$$

The margin is bounded as

$$s_{ij} = 8\,\tanh(f_{\text{edge}}(p_{ij}))\,,$$

and the binary relation logits are formed as

$$\ell_{ij} = [\,0,\ s_{ij}\,],$$

where the first channel represents a non-edge and the second channel represents an edge. The resulting relation tensor therefore has shape

$$[B, N, N, 2].$$

For molecular datasets, the same contextual decoder backbone is used. In addition to the edge-existence margin, the decoder contains a four-class bond-type head operating on the pair state,

$$f_{\text{bond}} : \mathbb{R}^{128} \to \mathbb{R}^4.$$

The edge-existence channel and four bond-type channels form a five-class pairwise molecular relation representation. Additional node-level heads predict molecular attributes including atom type, formal charge, hydrogen count, and aromaticity from the 32-dimensional node latent.

Table 10: Dataset-specific Stage-I decoder outputs.

Dataset	Relation classes	Bond-type head	Node-attribute heads	Realization
Planar / Tree	2	–	–	Argmax
SBM	2	–	–	Argmax
Protein	2	–	–	Argmax
MOSES	5	$128 \to 4$	Atom / charge / H / aromatic	Argmax
GuacaMol	5	$128 \to 4$	Atom / charge / H / aromatic	Argmax

Graph realization. Relations are realized directly from the decoder logits using

$$\hat{r}_{ij} = \arg\max_r \ell_{ij,r}.$$

The Stage-I models reported in this work do not use a graph-budget head, Top-K edge selection, Bernoulli edge sampling, threshold search, or post-hoc degree calibration. Thus, graph realization is determined directly by the contextual decoder predictions.

B.3 Stage-II Node-Field Transport Configuration

After Stage I, the encoder and decoder are frozen and Flow Matching is trained directly on the node field. Across all datasets, the persistent generative state contains only

$$Z \in \mathbb{R}^{N\times 32}$$

and a binary node-validity mask. No pair, edge, or graph-level latent state is transported in Stage II.

Latent normalization. Before Flow Matching, each latent dimension is standardized using statistics computed exclusively from valid node latents in the training split. Let $\mu, \sigma \in \mathbb{R}^{32}$ denote the resulting mean and standard deviation. We transform

$$x_1 = \frac{Z - \mu}{\sigma},$$

where the population standard deviation is used and each component of σ is lower-bounded by 10^{-6}. Padding nodes are excluded from the statistics and are set to zero after normalization. After sampling, the transformation is inverted once,

$$Z = x_1 \odot \sigma + \mu,$$

before applying the frozen Stage-I decoder.

Flow Matching formulation. The source field is sampled from an isotropic Gaussian,

$$x_0 \sim \mathcal{N}(0, I),$$

and masked so that padded node rows remain zero. We use the straight interpolation path

$$x_t = (1-t)x_0 + tx_1.$$

Training times are sampled according to

$$t = 0.98\,(0.02u + 0.98\,\mathrm{sigmoid}(\epsilon)) + 0.01,$$

where

$$u \sim \mathcal{U}(0,1), \qquad \epsilon \sim \mathcal{N}(0,1).$$

Thus, $t \in [0.01, 0.99]$ and is not sampled uniformly.

The transport network predicts the clean endpoint directly,

$$\hat{x}_1 = F_\theta(x_t, t, M, c),$$

where M is the node-validity mask and c is the self-conditioning field. The objective is

$$\mathcal{L}_{\text{transport}} = \frac{\sum_{i,k} M_i \left(\frac{x_{1,ik} - \hat{x}_{1,ik}}{\max(1-t, 0.05)}\right)^2}{32 \sum_i M_i}.$$

The denominator weighting makes the endpoint-prediction objective equivalent to matching the corresponding velocity residual. No auxiliary structural, pair-level, consistency, or KL loss is used in Stage II.

Table 11: Stage-II node-field transport architectures.

Parameter	Planar/Tree/SBM/Protein	MOSES/GuacaMol
Model	NodeX1DiT	NodeX1DiTBig
Input/output latent dim	32	32
Self-conditioning dim	32	32
Input projection	$64 \to 256$	$64 \to 768$
Hidden dimension	256	768
Transformer blocks	8	12
Attention heads	8	12
Head dimension	32	64
MLP ratio	4	4
FFN dimension	1024	3072
Normalization	LayerNorm	LayerNorm
LayerNorm ϵ	10^{-5}	10^{-5}
AdaLN modulation	6-way	6-way
MLP activation	GELU	GELU
Conditioning activation	SiLU	SiLU
Attention dropout	0.0	0.0
MLP dropout	0.0	0.0
Time embedding input dim	256	256
Node-count conditioning	Yes	Yes
Positional embedding	None	None
Graph conditioning	None	None
Prediction target	Endpoint x_1	Endpoint x_1
Parameters	$\sim$9.7M	$\sim$129.0M

Node-field DiT. The current node field and self-conditioning field are concatenated before the input projection,

$$[x_t, c] \in \mathbb{R}^{N \times 64}.$$

When self-conditioning is absent, c is replaced by zeros.

The DiT uses pre-normalized Transformer blocks with six-way AdaLN modulation. The scalar time embedding is combined with a graph-size embedding

$$\left[\frac{n}{256}, \frac{\log n}{\log 256}\right],$$

which is projected to the model hidden dimension and added to the time conditioning vector. No graph-level latent conditioning, node-index positional embedding, or pairwise attention bias is used.

Two DiT sizes are used, as summarized in Table 11.

All Transformer blocks use scaled dot-product self-attention. The final representation is normalized and mapped directly back to the 32-dimensional node field through a single linear output projection.

Optimization. Unless otherwise specified, Stage-II models are optimized with AdamW using

$$\text{lr} = 2 \times 10^{-4}, \qquad (\beta_1, \beta_2) = (0.9, 0.95), \qquad \text{weight decay} = 10^{-3}.$$

We clip the gradient norm at 1.0 and train in FP32. Self-conditioning is used with probability $p_{\text{sc}} = 0.5$ for all datasets.

Exponential moving average (EMA) weights are maintained throughout training with decay 0.9999. During the first 1000 optimizer steps, the EMA parameters are synchronized directly with the online model; exponential averaging begins afterwards. All reported sampling results use the EMA model. Table 12 summarizes the dataset-specific settings; Batch denotes the effective global batch size.

B.4 Self-Conditioning and Sampling Protocol

Training-time self-conditioning. We use self-conditioning with probability

$$p_{\text{sc}} = 0.5.$$

Table 12: Dataset-specific Stage-II optimization settings.

Dataset	Batch	Training budget	Warmup	DiT	p_{sc}
Planar / Tree	128	200K	0	8L/8H/256	0.5
SBM	128	100K	0	8L/8H/256	0.5
Protein	8	400K	0	8L/8H/256	0.5
MOSES / GuacaMol	128	600K	4K	12L/12H/768	0.5

A single Bernoulli decision is sampled for each mini-batch. When self-conditioning is activated, the model first produces an endpoint estimate without conditioning,

$$\widetilde{x}_1 = \mathrm{sg}\left[F_\theta(x_t, t, M, 0)\right],$$

where sg denotes stop-gradient. The same model is then evaluated a second time at the same state and time,

$$\hat{x}_1 = F_\theta(x_t, t, M, \widetilde{x}_1).$$

The preliminary prediction is computed without gradients, and the transport loss is applied only to the second prediction. When self-conditioning is not activated, the conditioning field is set to zero.

Importantly, the two passes use exactly the same x_t, t, and validity mask; self-conditioning therefore supplies an endpoint estimate rather than an additional noisy observation.

Sampling-time self-conditioning. Inference uses a different feedback procedure from training. Let x_k denote the current node field and let c_k denote the conditioning field. At the first solver step,

$$c_0 = 0.$$

Each subsequent step performs only one model evaluation,

$$\hat{x}_{1,k} = F_\theta(x_k, t_k, M, c_k),$$

and carries the current endpoint estimate forward,

$$c_{k+1} = \mathrm{sg}(\hat{x}_{1,k}).$$

Thus, sampling uses one-step-lagged endpoint feedback: the conditioning field at step $k+1$ is the endpoint prediction obtained at step k. Unlike training, no stochastic self-conditioning decision or same-state second forward pass is used during sampling.

ODE solver. We integrate the endpoint-induced velocity field with a fixed-step explicit Euler solver using 100 function evaluations. The time grid is

$$t_k = 0.01 + k\Delta t, \qquad \Delta t = \frac{0.98}{100}, \qquad k = 0, \ldots, 99.$$

At each step, the predicted endpoint is converted to a velocity as

$$v_k = \frac{\hat{x}_{1,k} - x_k}{\max(1 - t_k, 0.05)},$$

and the field is updated according to

$$x_{k+1} = x_k + \Delta t\, v_k.$$

The validity mask is reapplied after every update so that padded rows remain zero. The final solver time is $t_{99} = 0.9802$; no additional stochastic term is injected after initialization.

Because sampling-time self-conditioning reuses the previous endpoint estimate, each Euler step requires only one model evaluation. Therefore, a 100-step sample uses 100 NFEs rather than 200.

EMA sampling. Sampling and evaluation use EMA parameters with decay 0.9999. Given a fixed initial Gaussian field, the ODE trajectory is deterministic. After the final Euler step, the sampled field is denormalized using the training-set statistics and decoded once by the frozen Stage-I decoder.

B.5 Protein Latency Measurement Protocol

Table 13 lists the sampling configuration used for each method in the single-graph Protein latency comparison of Table 4. All methods run with batch size 1 in FP32 on a single RTX 4090; runtimes are averaged over five pre-specified seeds.

Table 13: Protein latency sampling configurations.

Method	**Sampling configuration**	**Notes**
DiGress	1000 reverse steps	–
AutoGraph	autoregressive, `top_k=40`, `max_length=11000`	–
LG-Flow	official Protein Flow-100 + self-conditioning + decoder	runtime only; no valid Protein quality result
GRAFT	node-field Flow-100 + self-conditioning + decoder	–

B.6 Sampling-Budget Alignment on Protein

Table 14 reports Protein generation quality for GRAFT at 50 and 100 NFEs over five random seeds. Increasing the sampling budget improves all four graph-statistic MMDs and reduces the overall Ratio from 7.33 ± 1.00 to 5.1 ± 0.46; the 100-NFE setting is used for the final generation-quality evaluation.

Table 14: Protein generation quality at 50 and 100 NFEs.

NFE	Deg.↓	Clus.↓	Orbit↓	Spec.↓	Ratio↓
50	<u>0.00136</u>	<u>0.06319</u>	<u>0.03899</u>	<u>0.00293</u>	<u>7.33 ± 1.00</u>
100	**0.00100**	**0.05140**	**0.02090**	**0.00250**	$\mathbf{5.1 \pm 0.46}$

B.7 Reproducibility Statement

An anonymized implementation of GRAFT is available at `https://anonymous.4open.science/r/GRAFT-D1A0`. Implementation details, hyperparameters, and sampling protocols are provided in the code.