MULTI-DOMAIN DISCRETE GRAPH GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Current graph generative models are overwhelmingly trained on a single input distribution and then sampled to reproduce it. In this work, we propose extending graph generation into multiple structural domains. Under our proposed multi-domain setting, existing single-domain models, even when given a domain label, sample every domain from a single shared internal distribution. In preliminary benchmarking, we demonstrate the existing model paradigm fails, even when state-of-the-art models are trained with multi-domain awareness. To address this, we propose Discrete-interpolative Graph Generation or DiG^2, a graph diffusion model specially-designed to internalize domain-conditioning along a bridged discrete path. Benchmarking demonstrates that DiG^2 achieves at least 85% sampling accuracy across all tested domains, while current domain-conditioned models suffer mode-collapse. Additionally, DiG^2 has the unique capacity for interpolating between domains and extrapolating to unseen structures. This interpolation mechanism uncovers targeted molecule generation for achieving comparable performance relative to a 38-times larger models. Anonymous Source Code: Available Here

1 INTRODUCTION

Graph generative models are overwhelmingly *single-domain*, current research is focused on developing models trained to reproduce one dataset per model (Vignac et al., 2023; Qin et al., 2025; Huang

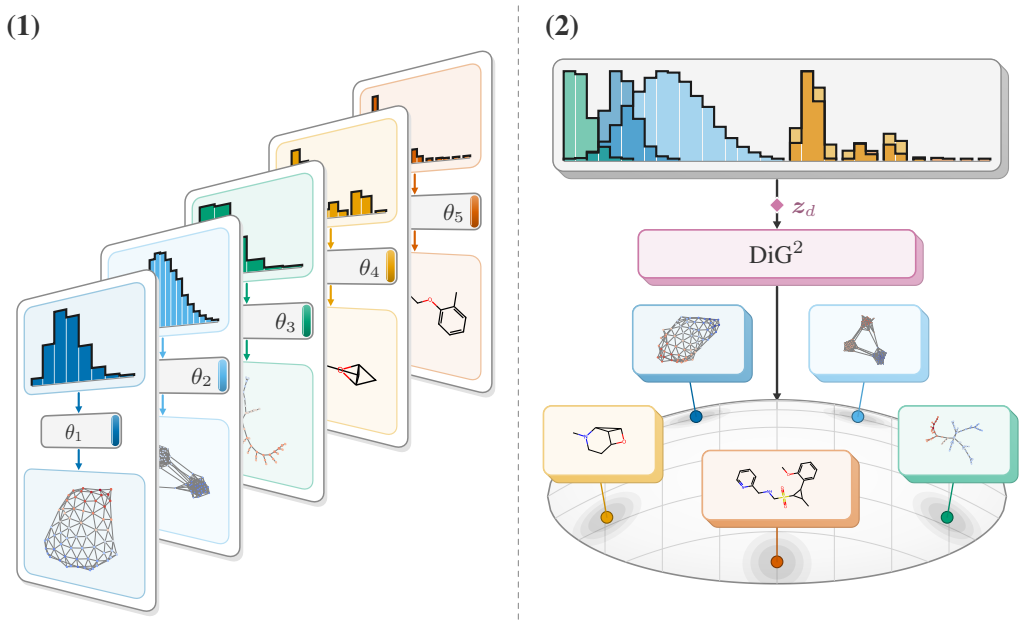


Figure 1: **One model per domain versus DiG^2.** Planar, SBM, Tree, QM9, ZINC. **(1)** Today, separate models $\theta_1, \ldots, \theta_5$ learn one distribution to re-create their input graphs. **(2)** DiG^2 reads multiple domains at once, generating from a single model, as tagged by a learned token z_d.

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.

& Birdal, 2026). This modeling paradigm produces strong results within a target domain, enabling reconstruction of difficult structures, such as planar graphs or drug-like molecules (Lee & Schachter, 1980; Brown et al., 2019). Outside of generation, graph learning is moving towards one model for many domains. Multi-domain graph foundation models train a single network across heterogeneous graph families and transfer between them (Sun et al., 2026; Wang et al., 2025a; 2026). However, graph generation has no such counterpart yet. We find that building a multi-domain graph generative model has two hurdles not present for generative models within the image and text domain.

The first challenge is proper *size and ordering*. Image and text generation operate on fixed-length output representations, such as pixel grids or token sequences, enabling consistent learning objectives and memory pressure (Austin et al., 2021; Radford et al., 2018). In graphs, node counts vary across and within domains. So, any model must manage differing combinatorial *and* canonical orderings between datasets (You et al., 2018). The second challenge concerns *vocabulary*. Image and text modalities share pixel values or token vocabularies (Yu et al., 2015; Devlin et al., 2019), reducing risk of leakage within transfer settings. In contrast, graph domains are semantically *disjoint*. A molecular double-bond has no counterpart among the edge classes of a stochastic block model (Ramakrishnan et al., 2014; Abbe, 2018; Irwin et al., 2012; Polykovskiy et al., 2020).

Existing multi-domain graph generation work sidesteps node and edge vocabulary entirely, learning directly from adjacency matrix structure (Wang et al., 2025b). Separately, multi-domain learning without graph generation indicates promise applying domain labels (Yuan et al., 2025) or graphon structure (Yuan et al., 2026) to transfer performance across graph domains. Despite this success, our initial investigation found key limitations (App. G). A generative model conditioned on graph-level domain labels makes the model domain-*aware*, but remains incapable of generating domain-*separable* samples. The lack of vocabulary-aware, multi-domain graph generative models leaves an open question. *Can a graph model successfully generate several structured domains at once?*

In order to answer this question, we propose extending the graph generation problem to consist of *multiple structural domains with distinct vocabularies*. This is currently unprecedented, given that LGGM is structure only and state-of-the-art graph generative models have only been tested on single domains, as shown in Figure 1(1). To overcome this, we introduce the Discrete-Interpolative Graph Generation model, or DiG^2, as shown in Figure 1(2). DiG^2 achieves successful multi-domain graph generation through a specialized domain conditioning block coupled with bridge denoising between explicit per-domain priors.

Contributions.

1. We develop the first structurally-distinct multi-domain graph generation benchmark, enabling evaluation of current graph generative models to function within the multi-domain graph generation problem (Section 4.1).
2. We propose DiG^2, a novel discrete bridge-diffusion model designed for multi-domain graph generation (Section 3).
3. We analyze the dynamics of multi-domain graph generation, centered around: benchmarking the capabilities of existing models and DiG^2 on difficult domains (Section 4.1), determining which components make DiG^2 effective at multi-domain graph generation (Section 4.2), and uncovering DiG^2's ability for de novo molecular generation (Section 4.3).

2 Background

Adjacency-based graph generation. Seminal work with GraphRNN (You et al., 2018) applied state transitions to autoregressively generate graphs from adjacency representations. More recently, G^{2}PT incorporated tokenized graph representations for powerful autoregressive generation (Chen et al., 2025). In parallel, diffusion emerged as the dominant paradigm for graph generation. GDSS (Jo et al., 2022) introduced continuous-time score-based generation by jointly perturbing node features and the adjacency matrix through a system of stochastic differential equations. DiGress (Vignac et al., 2023) instead formulated graph generation directly in the discrete domain, denoising categorical node and edge features under marginal transition chains derived from discrete diffusion (Austin et al., 2021). DeFoG (Qin et al., 2025) subsequently replaced the discrete diffusion chain with discrete flow matching (Lipman et al., 2023; Campbell et al., 2024), parameterizing transitions through

a continuous-time rate matrix. SPECTRE (Martinkus et al., 2022), LGD (Zhou et al., 2024), and GraphBSI (Petersen et al., 2026) all demonstrate promise for graph generation.

Structured and hybrid graph diffusion. Recent approaches introduce structural factorizations beyond one-shot representations. PARD (Zhao et al., 2024) combines autoregressive and diffusion by decomposing generation into blocks, where a shared model learns the conditional distribution over said blocks. HOG-Diff (Huang & Birdal, 2026) integrates higher-order graph skeletons into bridge diffusion, which guides generation from coarse structure toward pairwise connectivity. Despite advancements, prior approaches are predominantly domain-specific and do not train a single models to jointly capture heterogeneous graph domains.

Multi-domain graph generation. LGGM (Wang et al., 2025b) conditions diffusion-generated samples on an encoded natural-language prompt, which is broadcast onto the adjacency matrix. To the best of our knowledge, only LGGM has explicitly attempted generating graph samples from multiple domains with a single model. Our proposed study is distinct, we evaluate multi-domain graph generation with disjoint vocabularies between structural domains.

De novo molecule generation. Masked diffusion language models (Sahoo et al., 2024) and image diffusion (Rombach et al., 2022) obtain broad coverage over a shared vocabularies, enabling generalization transfer. Despite the difficulty for transfer in graphs, discrete diffusion has shown strong in-context learning (Lee et al., 2025; Liu et al., 2026) or even OOD-exploration (Lee et al., 2023) for *de novo* molecule generation.

Discrete Difusion for Multiple Domains. Current graph diffusion is trained from a shared noise distribution, predicting over one vocabulary. LGGM is multi-domain, but discards vocabulary to operate purely on the adjacency matrix. DiG^2 integrates domain conditioning at every denoiser layer, scoping predictions across each domain's vocabulary. Like HOG-Diff, DiG^2 samples along a bridge. However, DiG^2's sampler is specialized to disambiguate between a domain-*separable* graph and it's prior, overcoming the difficulty posed by a lack of domain-awareness (App. G).

3 Model Formulation

3.1 Preliminaries and Overview

Notation. A graph $\mathcal{G} = (\boldsymbol{X}, \mathbf{E})$ with n nodes has one-hot node classes $\boldsymbol{X} \in \{0,1\}^{n \times C_X}$ and one-hot edge classes $\mathbf{E} \in \{0,1\}^{n \times n \times C_E}$, where "no edge" is itself an edge class; x^i is the class of entry $i \in \mathbb{I}_n$ (a node or node pair). Every graph belongs to a domain $d \in \mathbb{D}$; here $\mathbb{D} = \{$Planar, SBM, Tree, QM9, ZINC$\}$. All domains share one global vocabulary of node and edge classes, but domain d only uses its subset $\mathbb{V}_d$ (i.e. bond types exist only in the molecular domains), $\boldsymbol{q}_d$, $\mathbb{V}_d$ and the embedding $\boldsymbol{z}_d$ are rows selected by the domain one-hot $\boldsymbol{e}_d$. Time runs from noise at $t = 0$ to data at $t = 1$, and $\mathcal{G}_t$ is a partially corrupted graph. The prior, the corruption of a given clean graph and the prediction $\boldsymbol{\pi}$ are categorical and sampled independently per entry.

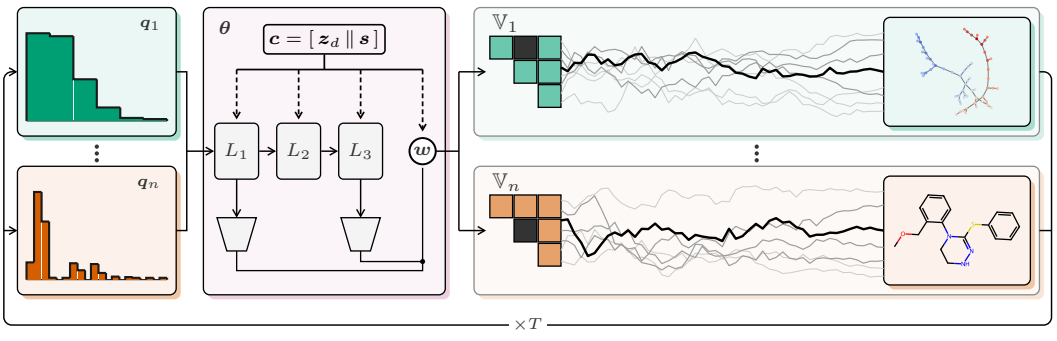


Figure 2: **DiG^2 generation.** A unique prior, $\boldsymbol{q}_d$, seeds a given domain (Section 3.2). A denoiser θ is domain-conditioned through every layer, a gate $\boldsymbol{w}$ mixes readouts into one prediction (Section 3.3). Each adjacency is masked over $\mathbb{V}_d$, restricting learning to relevant domains (Section 3.4). Entries are resolved from the masked $\boldsymbol{\pi}$ via a shared bridge toward a target domain (Section 3.5).

3.2 DOMAIN-CONDITIONED CORRUPTION AND ITS BRIDGE POSTERIOR

We define the denoising process over multiple graph domains, where corruption explicitly isolates a domain. To start, DiG^2 corrupts a clean graph $\mathcal{G}_1$ toward a *domain-specific* prior $\boldsymbol{q}_d$, rather than one shared endpoint. So, the amount of noise present at time t already reflects the domain d.

Definition 1 (Domain-conditioned corruption process)**.** *For a clean graph $\mathcal{G}_1$ and domain $d \in \mathbb{D}$ with prior $\boldsymbol{q}_d$, define the corruption process $\{P_t(\cdot \mid \mathcal{G}_1, d)\}_t$ by the entrywise linear interpolation,*

$$P_t(\mathcal{G}_t \mid \mathcal{G}_1, d) \;=\; \prod\nolimits_{i \in \mathbb{I}_n} \big[\, t\, \delta_{x_1^i} + (1-t)\, \boldsymbol{q}_d^i \,\big](x_t^i), \qquad \mathcal{G}_t \sim P_t(\cdot \mid \mathcal{G}_1, d). \tag{1}$$

At $t = 0$, $P_0(\cdot \mid \mathcal{G}_1, d) = \otimes_i \boldsymbol{q}_d^i$ is the domain prior; at $t = 1$, $P_1(\cdot \mid \mathcal{G}_1, d) = \delta_{\mathcal{G}_1}$ is the clean graph.

This generalizes domain-agnostic interpolation paths to $|\mathbb{D}|$ distinct endpoints indexed by domain.

Reverse sampling by direct interpolation. Generation runs Equation 1 backward from $\boldsymbol{q}_d$ to a clean graph. As a linear path, Equation 1's reverse transition is available in closed form:

Proposition 1 (Domain-conditional bridge sampling)**.** *Fix d, n and $0 \le t < s \le 1$, let $a = \frac{s-t}{1-t} \in (0, 1]$, and let $\pi_d^i(x \mid \mathcal{G}_t) = P(x_1^i{=}x \mid \mathcal{G}_t, d)$ under Definition 1. **(i) Shared kernel:** $P(\mathcal{G}_s \mid \mathcal{G}_t, \mathcal{G}_1, d) = \prod_i \big[a\, \delta_{x_1^i} + (1-a)\, \delta_{x_t^i}\big](x_s^i)$, which depends on neither d nor $\boldsymbol{q}_d$. **(ii) Exact step:** averaged over $\mathcal{G}_1 \sim P(\cdot \mid \mathcal{G}_t, d)$, maps $P_t(\cdot \mid d)$ to $P_s(\cdot \mid d)$ for any step size, so iterating it from $\mathcal{G}_0 \sim \otimes_i \boldsymbol{q}_d^i$ returns $P_{\mathrm{data}}(\cdot \mid d)$; entrywise,*

$$P(x_s^i \mid \mathcal{G}_t, d) \;=\; a\, \pi_d^i(\cdot \mid \mathcal{G}_t) + (1-a)\, \delta_{x_t^i}. \tag{2}$$

***(iii) Closure:** if $\operatorname{supp} \boldsymbol{q}_d^i \subseteq \mathbb{V}_d^i$, drawing entries from $a\, \boldsymbol{\pi}^i + (1-a)\, \delta_{x_t^i}$ with any $\boldsymbol{\pi}^i$ on $\mathbb{V}_d^i$, masked or guided, keeps every iterate in $\prod_i \mathbb{V}_d^i$.*

Per entry, Equation 2 is the row-stochastic matrix $\boldsymbol{B}^{(a)} = (1-a)\boldsymbol{I} + a\, \mathbf{1}(\boldsymbol{\pi}^i)^\top$, with no free parameters beyond a and $\boldsymbol{\pi}^i$. The domain thus acts only through π_d^i and $\mathbb{V}_d$, and the noisy state identifies the target domain. Precisely, Definition 1 is a mixture path of discrete flow matching from (Gat et al., 2024; Campbell et al., 2024) with a domain-specific source $\boldsymbol{q}_d$, and Proposition 1 is its exact finite-step reverse kernel. We call the resulting sampler a bridge because each domain's path is pinned to its own prior. Proof in App. A.1.

3.3 MULTI-DEPTH, DOMAIN-CONDITIONED DENOISER

Given that a 12-node molecule can be parametrized with fewer message-passing layers relative to a 144-node SBM graph (Gilmer et al., 2017), there is no one-size-fits-all architecture to balance learning on the two domains. This motivates development for a per-depth mechanism to optimize the required depth of denoiser prediction layers, activating deeper predictions only when necessary.

Multi-depth denoising. A shared denoiser $f(\cdot\,; \boldsymbol{\theta})$ is trained to invert Definition 1, conditioned on $\mathcal{G}_t$, t, and d. Instead of single predictions at the final layer, f exposes candidate reconstructions of $\mathcal{G}_1$ at several depths simultaneously,

$$\big\{\hat{\mathcal{G}}_1^{(\ell)}\big\}_{\ell \in \mathbb{K}} \;=\; f(\mathcal{G}_t, t, d\,; \boldsymbol{\theta}), \qquad \mathbb{K} \subset \{1, \dots, L\}, \tag{3}$$

where L is the number of transformer layers and $\mathbb{K}$ the set of readout depths. Readouts are combined within a lightweight gate $f_{\mathrm{gate}}(\cdot\,; \boldsymbol{\phi})$ conditioned on the same domain signal,

$$\hat{\mathcal{G}}_1 \;=\; \sum_{\ell \in \mathbb{K}} w_\ell(d)\, \hat{\mathcal{G}}_1^{(\ell)}, \qquad \boldsymbol{w}(d) = \operatorname{softmax}\big(f_{\mathrm{gate}}(d\,; \boldsymbol{\phi})\big). \tag{4}$$

f_{gate} is zero-initialized, so every domain starts from the uniform average $\boldsymbol{w}(d) = \frac{1}{|\mathbb{K}|}\mathbf{1}$ and learns its own weighting over depths during training.

Domain-conditioning block. Since our initial investigation determined that graph-level conditioning was too weak for multi-domain graph generation (Appendix G). We introduce a specialized domain-conditioning block, capable of discerning complicated graph structure and learning separation between input domains. The block maximizes domain-separable information within Equation 3,

entering through a conditioning vector $\boldsymbol{c} = [\boldsymbol{z}_d \| \boldsymbol{s}]$ in two parts. The first, $\boldsymbol{z}_d$ is a learned embedding table of domain labels, with a row reserved, $\boldsymbol{z}_\varnothing$, as the null label for classifier-free guidance (Section 3.5). A label states *which* domain is requested, but not *what its graphs look like*, and $\boldsymbol{s}$ supplies the latter. The second mechanism, $\boldsymbol{s}$ is a structural summary, obtained via a centered EMA prototype and message-passing. A small GNN $f_{\text{struct}}(\cdot\,; \boldsymbol{\omega})$ encodes the current noisy graph into $\boldsymbol{s}_{\text{cur}}$. The same encoder, applied to domain d's clean training graphs, maintains a prototype $\boldsymbol{s}_{\text{proto},d}$: an exponential moving average of their mean embedding, centered on the mean over domains and ℓ_2-normalized. Centering shared structure amongst domains, so the prototype encodes what distinguishes d. The two are then fused as $\boldsymbol{s} = f_{\text{fuse}}\big([\boldsymbol{s}_{\text{proto},d} \| \boldsymbol{s}_{\text{cur}}]\big)$. Critically, $\boldsymbol{c}$ is injected into the node and edge representations at *every* layer, rather than as a single graph-level vector alongside them. We provide an illustration of the conditioning block in Figure 5.

3.4 Training

Node vocabularies are often disjoint between domains. A molecule sample can have a carbon atom, which can join a triple bond. Whereas, nodes from a planar graph are connected exclusively by single edges. Meaning, a model trained on carbon atoms is likely to predict triple-bonds when attempting to construct a planar graph.

Domain-constrained training objective. Given this risk of vocabulary leakage, we restrict the cross-entropy objective for each graph to its own domain's valid class support $\mathbb{V}_d$,

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) \;=\; \mathbb{E}_{(\mathcal{G}_1, d) \sim P_{\text{data}}} \Big[\operatorname{CE}\big(\hat{\mathcal{G}}_1 \restriction_{\mathbb{V}_d},\, \mathcal{G}_1\big) \Big], \tag{5}$$

where $\restriction_{\mathbb{V}_d}$ masks the logits of classes outside $\mathbb{V}_d$ before the softmax. In effect, the model foregoes a gradient penalty while also avoiding probability assignment to classes outside a target domain. This constraint has no current analogue in single-domain discrete diffusion, all of which train with one shared vocabulary.

Training step. Domain-restriction upgrades training, where each step draws a graph and its domain, $(\mathcal{G}_1, d) \sim P_{\text{data}}$, a time $t = \psi(u)$ with $u \sim \operatorname{Unif}(0, 1)$ and $\psi(u) = 2u - u^2$, and a corrupted graph $\mathcal{G}_t$ from Equation 1. With probability p_{drop} the domain conditioning is replaced by the null conditioning $\boldsymbol{c}_\varnothing$, so the same network also learns the unconditional prediction used for guidance. Besides Equation 5 on the gated prediction, each non-terminal readout depth in $\mathbb{K}$ receives the same masked cross-entropy, averaged into a deep-supervision term $\mathcal{L}_{\text{deep-sup}}$ (App. C).

3.5 Sampling

A natural strength of diffusion models is classifier-free guidance (CFG) (Ho & Salimans, 2022). Currently, discrete graph diffusion is rightly-centered around CFG for property-targeting. Within this study, we treat target domains as a property, enabling domain-directed guidance exclusively via a 1-hot label.

Domain-directed guidance. The model evaluates once with the true domain label and once with a null label, giving masked logits $\boldsymbol{\ell}^c = F_{\boldsymbol{\theta}}(\mathcal{G}_t, t, \boldsymbol{c}) \restriction_{\mathbb{V}_d}$ and $\boldsymbol{\ell}^u = F_{\boldsymbol{\theta}}(\mathcal{G}_t, t, \boldsymbol{c}_\varnothing) \restriction_{\mathbb{V}_d}$. Both predictions are linearly extrapolated with a domain-specific strength $g(d)$,

$$\boldsymbol{\pi} \;=\; \operatorname{softmax}\big(\boldsymbol{\ell}^u + g(d)\,(\boldsymbol{\ell}^c - \boldsymbol{\ell}^u)\big). \tag{6}$$

The extrapolation is an expanded linear form of classifier-free discrete samplers, grounded in domain-specific prior distributions.

Sampling loop. This domain-awareness then allows generation to start from $\mathcal{G}_0 \sim \otimes_i \boldsymbol{q}_d^i$ and take T steps along the grid $t_k = \psi(k/T)$. At each step it computes $\boldsymbol{\pi}$ from Equation 6 and draws the next graph from Equation 2. Every entry is resolved to a draw from $\boldsymbol{\pi}^i$ with probability a and otherwise kept. The prediction stands in for π_d^i, with marginals in place of the joint posterior and a guidance tilt for $k < T - 1$ (App. A.1 quantifies both). The final step ($s = 1$) reads out the conditional prediction $\operatorname{softmax}(\boldsymbol{\ell}^c)$ rather than the guided $\boldsymbol{\pi}$, so guidance shapes the trajectory but not the last draw. Algorithms 1 and 2 give the full training and sampling procedures.

Training and sampling procedures. DiG^2 is trained by Algorithm 1, which runs Algorithm 3 once per batch, fine-tuned on the structural domains by Algorithm 4, and sampled by Algorithm 2.

$\boldsymbol{q}_d$ is domain d's normalized training class counts (nodes and node pairs), and $P(n \mid d)$ its training-size histogram. $F_{\boldsymbol{\theta}}$ (Section 3.3) returns one logit vector per entry, a node or an unordered node pair, and the per-depth readouts used by $\mathcal{L}_{\text{deep-sup}}$; $\restriction_{\mathbb{V}_d}$ is the mask described below, and $\text{CE} = \text{CE}_X + \lambda_E \text{CE}_E$ (Table 5). $\boldsymbol{\theta}$ includes the embedding table and f_{fuse}; all weights train with AdamW (AMSGrad, gradient norm clipped at 1.0). Algorithm 2 updates $\mathcal{G} = \mathcal{G}_{t_k}$ in place and rebuilds $\boldsymbol{c}$ from it at every call, as line 6 of Algorithm 1 does; $\boldsymbol{c}_\varnothing$ is Equation 16. The domain enters Algorithm 1 at lines 5–8 and Algorithm 2 at lines 1–2, 5 and 7–8 (line 8 is Equation 6); line 13 redraws every entry from $\boldsymbol{\pi}^i$ with probability a_k, at every step, and otherwise keeps it, a step shared by all domains (Proposition 1(i)).

Algorithm 1 DiG^2 training

Require: data P_{data}; weights $\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\omega}$
Ensure: trained $\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\omega}$; prototypes $\boldsymbol{s}_{\text{proto},d}$

1: **while** not converged **do**
2: draw a batch $B \sim P_{\text{data}}$
3: **for** each $(\mathcal{G}_1, d) \in B$ **do**
4: $u \sim \text{Unif}(0,1)$; $t \leftarrow \psi(u)$
5: $\mathcal{G}_t \sim P_t(\cdot \mid \mathcal{G}_1, d)$ ▷ noise to $\boldsymbol{q}_d$
6: $\boldsymbol{s} \leftarrow f_{\text{fuse}}([\boldsymbol{s}_{\text{proto},d} \| f_{\text{struct}}(\mathcal{G}_t)])$
7: $\boldsymbol{c} \leftarrow [\boldsymbol{z}_d \| \boldsymbol{s}]$, or $\boldsymbol{c}_\varnothing$ w.p. p_{drop}
8: $\hat{\mathcal{G}}_1 \leftarrow F_{\boldsymbol{\theta}}(\mathcal{G}_t, t, \boldsymbol{c}) \restriction_{\mathbb{V}_d}$
9: **end for**
10: $\mathcal{L} \leftarrow \text{CE}(\hat{\mathcal{G}}_1, \mathcal{G}_1) + \mathcal{L}_{\text{deep-sup}}$
11: write prototypes from B (Alg. 3)
12: $\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\omega} \leftarrow$ AdamW step on $\nabla\mathcal{L}$
13: **end while**

Algorithm 2 DiG^2 sampling

Require: domain d; steps T; $g(d)$; $F_{\boldsymbol{\theta}}$

1: $n \sim P(n \mid d)$ ▷ graph size
2: $\mathcal{G} \leftarrow \mathcal{G}_0$: $x^i \sim \boldsymbol{q}_d^i$ for $i \in \mathbb{I}_n$ ▷ prior
3: **for** $k = 0, \ldots, T-1$ **do**
4: $t_k, t_{k+1} \leftarrow \psi(k/T)$, $\psi((k+1)/T)$
5: $\boldsymbol{\ell}^c \leftarrow F_{\boldsymbol{\theta}}(\mathcal{G}, t_k, \boldsymbol{c}) \restriction_{\mathbb{V}_d}$
6: **if** $k < T-1$ **then**
7: $\boldsymbol{\ell}^u \leftarrow F_{\boldsymbol{\theta}}(\mathcal{G}, t_k, \boldsymbol{c}_\varnothing) \restriction_{\mathbb{V}_d}$
8: $\boldsymbol{\pi} \leftarrow \text{softmax}\big(\boldsymbol{\ell}^u + g(d)(\boldsymbol{\ell}^c - \boldsymbol{\ell}^u)\big)$
9: **else**
10: $\boldsymbol{\pi} \leftarrow \text{softmax}(\boldsymbol{\ell}^c)$ ▷ unguided
11: **end if**
12: $a_k \leftarrow (t_{k+1} - t_k)/(1 - t_k)$
13: redraw each $x^i \sim \boldsymbol{\pi}^i$ w.p. a_k
14: **end for**
15: **return** $\mathcal{G}$

4 Experiments

In order to assess the difficulty of the multi-domain graph generation problem and determine the effectiveness of DiG^2, we conduct experiments to answer the following research questions: **(RQ1):** Are domain-aware graph generative models and DiG^2 capable of generating multiple graph domains at once? **(RQ2):** What components of DiG^2 are necessary for successful multi-domain graph generation? **(RQ3):** Can DiG^2 interpolate for successful *de novo* molecule generation?

4.1 Main Results

Within the following section, we answer **RQ1** by benchmarking the capabilities of current graph generation models and DiG^2 to assess the difficulty of the multi-domain graph generation problem.

Setup We evaluate on five domains with disjoint semantics families: structural domains (Planar, SBM, Tree) and molecular domains (QM9, ZINC). All domains are encoded into a global vocabulary of 11 distinct node and 5 edge classes. This shared encoding makes single jointly-trained models possible, and makes domain conditioning failures decidable by inspection.

Arms. All benchmarked models are trained on the five dataset corpus under one shared schedule of 2.5M steps (10-day max training budget, Appendix D). DiG^2 is anchor fine-tuned, other models are fine-tuned, up to 25K steps on structural domains (Algorithm 4, Appendix C). 1-hot embeddings are projected into node- and edge-channels of single-domain models, as well as $\mathbb{V}_d$-mask sampling and per-domain priors enabling multi-domain probability output and conditioning, where applicable. LGGM integrates 1-hot labels and native text-prompt conditioning (Devlin et al., 2019). G2PT conditions natively on a per-domain prefix token. Every arm is re-parented onto a single configuration, so the dataset, splits, optimizer schedule and evaluation harness are identical across generative models. Backbones, parameter counts and method-specific hyperparameters are in Appendix D.

Metrics. For structural domains, we report: *Accuracy* on planarity, acyclicity, or block-structure and *average MMD Ratios* across degree, clustering, orbit, spectral and wavelet distributions. For molecular domains we report Validity and Fréchet ChemNet Distance (FCD) (Preuer et al., 2018). Models are chosen based on best average validation and accuracy metrics.

	Structural (Acc. ↑ / Ratio ↓)			Molecular (V.U.N. ↑ / FCD ↓)	
Model	Planar	SBM	Tree	QM9	ZINC
DiG^2	**100/ 2.3**	**85 / 1.8**	**97.5 / 1.56**	**99.92 / 0.29**	**95.46 / 2.7**
G2PT	2.5/ 7.5	73.1/4.92	87.4/1.57	90.6 / 0.55	72.6 / 6.4
DeFoG†	95.0/4.3	5.0/6.9	62.5/2.35	90.21 / 0.49	92.93 / 3.4
DiGress†	10.0/7.8	37.5/ 3.04	67.5/1.81	95.43 / 0.4	87.1 / 4.7
HOG-Diff†	0.0/ 29.5	25.0/ 4.72	52.5/3.73	93.20 / 1.5	86.38 / 4.0
GraphBSI†	0.0/ 30.1	22.5/ 3.56	72.5/1.95	43.50 / 9.4	42.19 / 25.2
LGGM	0.0/ 76	12.5/ 9.1	0.0 / 8.78	86.3 / 0.75	85.2 / 8.55
Pard†	0.0/ 298	–	47.5 / 4.96	47.48 / 10.1	31.4 / 25.2
GraphRNN†	0.0/ 156	10.0 / 22.58	22.5 / 2.76	47.53 / 10.5	32.83 / 25.6
LGD†	0.0/ 2591	0.0 / 253	0.0 / 375	34.2 / 19.1	37.89 / 26.1
SPECTRE†	0.0/ 2766	0.0 / 374	0.0 / 461	17.3 / –	9.51 / –

Table 1: Multi-domain generation results across two distinct structural families. † denotes a baseline given multi-domain conditioning uplift. Sampling is fixed to T=200 steps for all diffusion baselines.

DiG^2 succeeds at multi-domain generation. As shown in Table 1, DiG^2 achieves 100.0%, 85.0%, and 97.5% structural accuracy on PLANAR, SBM, and TREE, and reaches 99.92% validity on QM9 and 95.46% on ZINC, **performing the best on every metric**. The current benchmark therefore indicates generation across multiple domains requires sampling paths and internal representations to remain coordinated with the requested domain.

Domain conditioning does not solve multi-domain generation. In contrast, Table 1 also exposes a substantial gap for single-domain models, even after the models are provided domain-conditioning. For example, DeFoG retains 95.0% planarity accuracy, but falls to 5.0% on SBM. G^{2}PT is substantially stronger on SBM and TREE (73.1% and 87.4%, respectively), yet falls to 2.5% on PLANAR. More specialized graph generators are brittle: HOG-Diff, GraphBSI, PARD, GraphRNN, and LGD all fail to recover at least one structural family. SPECTRE suffers mode collapse under the joint training protocol. Therefore, domain conditioning alone is insufficient for enforcing domain-separable generative dynamics.

4.2 Ablation Study

To answer **(RQ2)** and isolate where separate DiG^2 components are necessary for multi-domain graph generation. We conduct a leave-one-component-out ablation beginning from the full DiG^2 configuration. All results can be found in Table 2.

Setup We ablate the following model components: (i) the EMA prototype embedding (Morales-Brotons et al., 2024), forcing the denoiser to rely on 1-hot conditioning alone; (ii) per-domain corruption priors, replacing q_d with one pooled prior; (iii) domain-restricted cross-entropy support, allowing logits over the full global vocabulary; and (iv) multi-depth readout, retaining only terminal-layer predictions. This isolates representation, endpoint, vocabulary-support, and computation-depth effects, respectively.

Results Table 2 demonstrates that *each of DiG^2's components are critical* for successful multi-domain graph generation. In particular, the EMA prototype has the strongest detrimental effect, with -22.5% PLANAR and -42.5% SBM accuracy, which is indicative of the difficulty for internal model representations to separate the two, even with the remaining per-domain components. Separately, the lack of complete collapse across every test suggests that Equation 2 maintains enough model capacity to resist mode collapse during the denoising process. In-depth analysis of separate components is provided in Appendix E.

DiG^2	Structural (Acc. ↑ / Ratio ↓)			Molecular (V.U.N. ↑ / FCD ↓)	
	Planar	SBM	Tree	QM9	ZINC
Full model (Table 1)	100/2.3	85 / 1.8	97.5 / 1.56	99.92 / 0.29	95.46 / 2.7
w/o EMA prototype	77.5/7.65	42.5 / 4.82	92.5 / 1.79	92.48 / 1.83	91.92 / 3.37
w/o per-domain priors	82.5/4.29	70.0 / 3.95	95.0 / 1.73	93.86 / 1.48	92.42 / 2.93
w/o per-domain CE	90.00/6.37	77.5 / 3.48	95.0 / 1.69	94.57 / 1.73	94.42 / 2.89
w/o per-layer depth	95.0/2.95	72.5 / 5.13	90.0 / 2.78	98.80 / 0.79	95.04 / 3.01

Table 2: Ablation Study Results. As shown, all of the following ablated mechanism are necessary for optimal multi-domain generation: per-domain corruption priors $\boldsymbol{q}_d$, support restriction $\restriction_{\mathbb{V}_d}$, multi-depth readout from Figure 2, and the EMA embedding in Figure 5.

4.3 α-INTERPOLATION FOR ZERO-SHOT MOLECULAR EXTRAPOLATION

The previous studies test whether generation modes remain separated. However, this study is designed to answer **RQ3**. A complementary question on whether DiG^2's learned modes can be interpolated to generate novel, yet realistic samples, for successful *de novo* molecule generation.

α-Interpolation. In order to determine if DiG^2 can traverse the space between domains to generate unseen molecular corpora. We treat DiG^2's domain conditioning with a specialized form of classifier-free guidance. For $\alpha \in [0, 1]$, with $\alpha = 0$ denoting QM9 and $\alpha = 1$ denoting ZINC, we construct an interpolated corruption endpoint and conditioning state,

$$z_\alpha = (1 - \alpha)\, z_{\mathrm{QM9}} + \alpha\, z_{\mathrm{ZINC}}, \tag{7}$$

$$s_{\mathrm{proto},\alpha} = (1 - \alpha)\, s_{\mathrm{proto,QM9}} + \alpha\, s_{\mathrm{proto,ZINC}}, \tag{8}$$

and assemble $c_\alpha = [z_\alpha \| s_{\mathrm{proto},\alpha}]$. Sampling then uses the same closed-form reverse interpolation as Prop. 1, but masks logits to the union molecular support $V_\alpha = V_{\mathrm{QM9}} \cup V_{\mathrm{ZINC}}$, held fixed across α. No GuacaMol or MOSES graph is used for parameter updates.

Given the pinned α endpoint, small t for every molecule resembles the prior. So, blended priors make interpolation apparent via sampled distribution. QM9 and ZINC provide a controlled setting, both datasets are composed of similar vocabularies, differing mostly in size and chemical distribution. Therefore, any extrapolated molecule close to a given target tests how DiG^2's α-interpolation discovers viable molecules outside it's training.

Baselines. We compare DiG^2 against three molecular generators which we treat to expose a QM9↔ZINC axis. LGGM (Wang et al., 2025b) is given a per-domain one-hot prompt table, and α blends the two embeddings. GenMol (Lee et al., 2025), a discrete molecule generation model, fine-tuned with QM9 and ZINC domain tokens for $12,000$ steps, and α blends their embeddings. DemoDiff (Liu et al., 2026), a 700M-parameter molecule generation foundation model. We condition DemoDiff on demonstration molecules drawn from QM9 and ZINC in proportion $(1 - \alpha) : \alpha$. Given space limitations, tests with LGGM and further details are provided in Appendix F.

Zero-shot interpolation. Every model is evaluated on the same fixed grid, $\alpha \in \{0, 0.5, 0.75, 0.85, 0.95, 1\}$, three seeds per cell and 2,000 attempts, on GuacaMol molecules of 10–14 heavy atoms. Target metrics never tune parameters or guidance.

Despite Figure 3 being confined to $\alpha \in [0, 1]$, DiG^2 interpolates between both the $p25$ and $p50$ QED target, while DemoDiff can only reach the $p50$ target and GenMol overshoots it (Bickerton et al., 2012). This lead to the question of, could DiG^2 *extrapolate* to discover realistic samples from a wider α target?

4.4 EXTERIOR α: INTERPOLATE-TO-EXTRAPOLATE

Interpolate-to-Extrapolate. Given that α-interpolation is a row blend, it is defined for $\alpha > 1$, which extrapolates along the QM9→ZINC direction. Neither baseline admits this, as DemoDiff's α is a mixture proportion and GenMol is inflexible. As shown in Figure 4, extending the MOSES 20–27 sweep requires the interpolation to traject past ZINC's $a = 1$ endpoint. The p75 logP target is met only at interpolation values of $\alpha^* > 1$. DiG^2 covers 10 times the logP interpolation range of De-

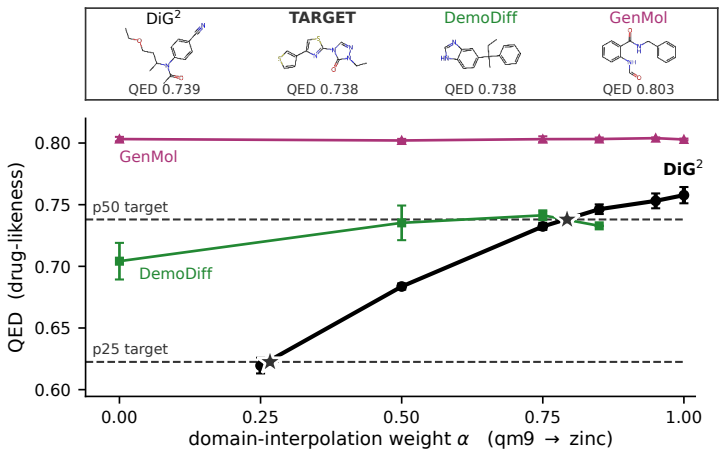


Figure 3: **QED as an interpolation target.** GuacaMol 10-14 band. QED range covered: DiG^2 = 0.138, DemoDiff = 0.037, GenMol = 0.002, meaning GenMol's axis is inert, and DemoDiff's moves but cannot descend to p25. Per-cell values are in Table 12.

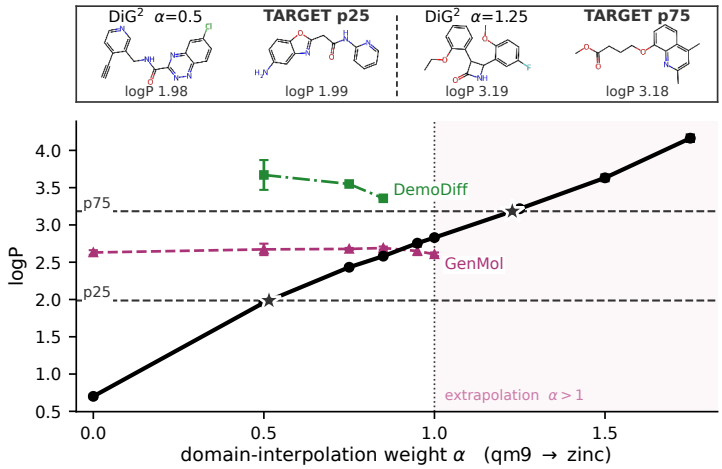


Figure 4: **logP as an extrapolation target.** MOSES 20-27 band. Each arm is swept through *its own* native mechanism, as in Figure 3. logP range covered: DiG^2 = 3.46, DemoDiff = 0.31 GenMol = 0.08, indicating GenMol and DemoDiff are inert for p25 and p75 targets. Per-cell values are in Table 13.

moDiff, indicating internal model capacity to extrapolate onto molecular distributions well outside it's original training distribution. Raw values for Filters, SNN, and Validity are in Appendix F.

5 Conclusion

Within this work, we formalize the multi-domain graph generation problem as a challenging task, requiring the separation of internal representations from disjoint vocabularies and generating domain-specific structure. To address this problem, we introduce DiG^2, a novel graph diffusion model integrating domain-conditioning to bridge between domain-specific priors for successful multi-domain graph generation. We also uncover DiG^2's ability to $\alpha-$interpolate across domains, enabling the model to reach outside the scope of it's original training distribution and generate novel molecules. We believe the multi-domain graph generation problem is a promising frontier for generative modeling, enabling future work for exciting applications of graph generative systems.

AI USE STATEMENT

In this work, we used generative AI tools to assist in the writing of proofs. We have not used generative AI tools for: idea formulation, help developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, proposing or refining hypotheses, designing feedback on research methodology or experiments, implementing methods, assisting with translation, cleaning and reformatting data, supporting qualitative and thematic data analysis, and interpreting results. Using an LLM to generate synthetic data sets is not applicable to this work. Additionally, we used generative AI tools for: drafting parts of this research paper, editing software code, and modifying scientific figures. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness by the first author, mathematical proofs were refined by 2 authors and checked by all authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Given the nature of graph generation research, we acknowledge that advancing this subfield could have unintended consequences for generative models to be applied for dual-use.

REPRODUCIBILITY STATEMENT

We have included the following in order to promote the reproducibility of our work: implementation details indicating exact hyperparameters (Appendix C), anonymous source code here and within the abstract, and deeper details on our theoretical proof (Appendix A.1).

REFERENCES

Emmanuel Abbe. Community detection and stochastic block models: recent developments. *Journal of Machine Learning Research*, 18(177):1–86, 2018.

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

G Richard Bickerton, Gaia V Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90–98, 2012.

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), 2019.

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. *arXiv preprint arXiv:2402.04997*, 2024.

Xiaohui Chen, Yinkai Wang, Jiaxing He, Yuanqi Du, Soha Hassoun, Xiaolin Xu, and Liping Liu. Graph generative pre-trained transformer. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 9176–9197. PMLR, 13–19 Jul 2025. URL `https://proceedings.mlr.press/v267/chen25bl.html`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. In *Advances in Neural Information Processing Systems*, 2024.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. Pmlr, 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. URL https://arxiv.org/abs/2207.12598.

Yiming Huang and Tolga Birdal. Hog-diff: Higher-order guided diffusion for graph generation. In *International Conference on Learning Representations*, pp. 111572–111614, 2026.

John J Irwin, Teague Sterling, Michael M Mysinger, Erin S Bolstad, and Ryan G Coleman. Zinc: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52 (7):1757–1768, 2012.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International conference on machine learning*, pp. 10362–10383. PMLR, 2022.

Der-Tsai Lee and Bruce J Schachter. Two algorithms for constructing a delaunay triangulation. *International Journal of Computer & Information Sciences*, 9(3):219–242, 1980.

Seul Lee, Jaehyeong Jo, and Sung Ju Hwang. Exploring chemical space with score-based out-of-distribution generation. In *International Conference on Machine Learning*, pp. 18872–18892. PMLR, 2023.

Seul Lee, Karsten Kreis, Srimukh Prasad Veccham, Meng Liu, Danny Reidenbach, Yuxing Peng, Saee Paliwal, Weili Nie, and Arash Vahdat. Genmol: A drug discovery generalist with discrete diffusion. *arXiv preprint arXiv:2501.06158*, 2025.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Gang Liu, Jie Chen, Yihan Zhu, Michael Sun, Tengfei Luo, Nitesh Chawla, and Meng Jiang. Graph diffusion transformers are in-context molecular designers. In *International Conference on Learning Representations*, pp. 101812–101846, 2026.

Liheng Ma, Chen Lin, Derek Lim, Adriana Romero-Soriano, Puneet K Dokania, Mark Coates, Philip Torr, and Ser-Nam Lim. Graph inductive biases in transformers without message passing. In *International Conference on Machine Learning*, pp. 23321–23337. PMLR, 2023.

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 (ICML)*, 2022. arXiv:2204.01613.

Daniel Morales-Brotons, Thijs Vogels, and Hadrien Hendrikx. Exponential moving average of weights in deep learning: Dynamics and benefits. *Transactions on Machine Learning Research*, 2024.

Ole Petersen, Marcel Kollovieh, Marten Lienen, and Stephan Günnemann. Discrete bayesian sample inference for graph generation. In *International Conference on Learning Representations*, volume 2026, pp. 83731–83757, 2026.

Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, et al. Molecular sets (moses): a benchmarking platform for molecular generation models. *Frontiers in pharmacology*, 11:565644, 2020.

Kristina Preuer, Philipp Renz, Thomas Unterthiner, Sepp Hochreiter, and Günter Klambauer. Fréchet ChemNet distance: A metric for generative models for molecules in drug discovery. *Journal of Chemical Information and Modeling*, 58(9), 2018.

Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. DeFoG: Discrete flow matching for graph generation. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 50269–50326. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/qin25d.html.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1, 2014.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 10674–10685. ieee, 2022.

Subham S Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

Li Sun, Zhenhao Huang, Silei Chen, Lanxu Yang, Junda Ye, Sen Su, and Philip Yu. Multi-domain riemannian graph gluing for building graph foundation models. In *International Conference on Learning Representations*, pp. 37129–37169, 2026.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=UaAD-Nu86WX.

Shuo Wang, Bokui Wang, Zhixiang Shen, Boyan Deng, and Zhao Kang. Multi-domain graph foundation models: Robust knowledge transfer via topology alignment. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 64806–64821. PMLR, 13–19 Jul 2025a. URL https://proceedings.mlr.press/v267/wang25dj.html.

Yu Wang, Ryan Rossi, Namyong Park, Huiyuan Chen, Nesreen Ahmed, Puja Trivedi, Franck Dernoncourt, Danai Koutra, and Tyler Derr. A large-scale training paradigm for graph generative models. In *International Conference on Learning Representations*, pp. 23212–23240, 2025b.

Zehong Wang, Zheyuan Zhang, Tianyi Ma, Chuxu Zhang, and Yanfang Ye. Generative graph pattern machine. *Advances in Neural Information Processing Systems*, 38:30068–30091, 2026.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pp. 5708–5717. PMLR, 2018.

Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

Haonan Yuan, Qingyun Sun, Junhua Shi, Xingcheng Fu, Bryan Hooi, Jianxin Li, and Philip S Yu. How much can transfer? bridge: Bounded multi-domain graph foundation model with generalization guarantees. In *Forty-second International Conference on Machine Learning*, 2025.

Haonan Yuan, Qingyun Sun, Junhua Shi, Xingcheng Fu, Bryan Hooi, Jianxin Li, and Philip S Yu. Graver: Generative graph vocabularies for robust graph foundation models fine-tuning. *Advances in Neural Information Processing Systems*, 38:16639–16684, 2026.

Lingxiao Zhao, Xueying Ding, and Leman Akoglu. Pard: Permutation-invariant autoregressive diffusion for graph generation. *Advances in Neural Information Processing Systems*, 37:7156–7184, 2024.

Cai Zhou, Xiyuan Wang, and Muhan Zhang. Unifying generation and prediction on graphs with latent graph diffusion. *Advances in Neural Information Processing Systems*, 37:61963–61999, 2024.

A Appendix

A.1 Proofs

Roadmap. The following proof answers four questions about the sampler of Section 3.2 and Section 3.5, in order. *(i) Is the reverse step exact, and is it really shared by all domains?* Lemmas A.1–A.3 build up to the proof of Proposition 1. *(ii) Where can the domain enter the prediction, and how much does the noisy graph already reveal about it?* Lemma A.4 and Table 4. *(iii) Are vocabulary masking and guidance safe to combine with the bridge?* Lemma A.5 and Corollary A.6, which justify how Equation 6 is implemented. *(iv) What does it cost to replace the exact posterior with a network?* Proposition A.7. Most proofs are short because a single picture, the reveal coupling of Lemma A.1, does the work; we state it first and refer back to it throughout.

Setup and premises. A graph with n nodes is the collection of classes $\mathcal{G} = (x^i)_{i \in \mathbb{I}_n}$ over its entries $\mathbb{I}_n = [n] \cup \{\{u, v\} : 1 \le u < v \le n\}$, of which there are $N_n = n + \binom{n}{2}$. A node entry takes values in the global node vocabulary (11 classes) and a pair entry in the global edge vocabulary (5 classes, including "no edge"). The node count n is drawn from domain d's size distribution before $t = 0$ and held fixed, so every statement below is conditional on (d, n). The domain reaches the sampler only through rows of tables selected by the same one-hot $\boldsymbol{e}_d \in \{0,1\}^{|\mathbb{D}|+1}$, whose last coordinate is the guidance null token $\varnothing$ (Table 3): the prior $\boldsymbol{q}_d^i = \mathbf{Q}_{\mathrm{pr}}^\top \boldsymbol{e}_d$ (its node or edge row), the vocabulary mask $\boldsymbol{m}_d^i = \mathbf{M}_{\mathrm{voc}}^\top \boldsymbol{e}_d$ with support $\mathbb{V}_d^i = \{x : \boldsymbol{m}_d^i(x) = 1\}$, and the embedding $\boldsymbol{z}_d = \mathrm{MLP}(\boldsymbol{E}^\top \boldsymbol{e}_d)$ inside $\boldsymbol{c} = [\boldsymbol{z}_d \| \boldsymbol{s}]$. We use three premises, all of which hold in our implementation:

(A1) $\operatorname{supp} \boldsymbol{q}_d^i \subseteq \mathbb{V}_d^i$: the prior lies inside the vocabulary;

(A2) $\operatorname{supp} P_{\mathrm{data}}(\cdot \mid d) \subseteq \prod_i \mathbb{V}_d^i$: $\mathbb{V}_d$ contains every class domain d uses;

(A3) both guidance branches are masked with the same $\boldsymbol{m}_d$ before they are combined.

Each premise closes one gap. Without (A1), the initial graph $\mathcal{G}_0$ could already contain a class that domain d never uses, and no later step would be obliged to remove it. Without (A2), the mask would forbid classes that real data of domain d contains, so the exact posterior would place mass where the masked network cannot, and masking would add error instead of removing it. Without (A3), guidance can undo or even invert a mask applied to only one branch (Lemma A.5(c)). We write $\Pi_d(\mathcal{G}_1 \mid \mathcal{G}_t) = P(\mathcal{G}_1 \mid \mathcal{G}_t, d)$ for the exact posterior of Definition 1, $\pi_d^i(x \mid \mathcal{G}_t) = P(x_1^i = x \mid \mathcal{G}_t, d)$ for its entry marginals, and $B_{t \to s}(\cdot \mid x_t, x_1) = a\, \delta_{x_1} + (1 - a)\, \delta_{x_t}$, with $a = a(t, s) = \frac{s-t}{1-t}$, for the conditional bridge kernel of one entry. Π_d does not factorize over entries; the prior, the corruption given $\mathcal{G}_1$, and the network's prediction do.

Table 3: Where the domain one-hot $\boldsymbol{e}_d$ enters the sampler, and what each entry point guarantees.

Object	From $\boldsymbol{e}_d$	Enters	Guarantee
prior $\boldsymbol{q}_d = \mathbf{Q}_{\mathrm{pr}}^\top \boldsymbol{e}_d$	fixed row	$\mathcal{G}_0$; the likelihood inside Π_d	the state carries evidence of d (Lemma A.4)
mask $\boldsymbol{m}_d = \mathbf{M}_{\mathrm{voc}}^\top \boldsymbol{e}_d$	fixed row	support of both guidance branches	closure (Prop. 1(iii)); non-expansive; survives guidance (Lemma A.5)
embedding $\boldsymbol{z}_d$, prototype $\boldsymbol{s}_{\mathrm{proto},d}$	learned row	$\boldsymbol{\pi}$, through $\boldsymbol{c}$	supplies d without inferring it from the state (Lemma A.4(c))
guidance weight $g(d)$	lookup	tilt of $\boldsymbol{\pi}$	tempers $P(d \mid x_1^i, \mathcal{G}_t)$ (Corollary A.6)
bridge kernel $B_{t \to s}$	none	every transition	shared by all domains (Prop. 1(i))

The reveal picture. Equation 1 says that at time t each entry shows its clean value with probability t and a prior draw otherwise. A convenient way to produce exactly this is to give every entry its own hidden alarm clock τ^i, uniform on $[0,1]$: before the clock rings the entry displays a fixed noise class x_0^i drawn from the prior, and from the moment it rings it displays its clean class x_1^i. Time t then simply reveals the entries whose clocks have rung. Lemma A.1 checks that this construction reproduces Equation 1; its value is that questions about the reverse step become questions about when independent clocks ring.

Lemma A.1 (Reveal coupling)**.** *Draw $\mathcal{G}_0 \sim \otimes_i \boldsymbol{q}_d^i$, $\mathcal{G}_1 \sim P_{\mathrm{data}}(\cdot \mid d)$ and reveal times $\tau^i \sim \mathrm{Unif}(0,1)$, all independent, and for $u \in [0,1]$ set $x_u^i = x_1^i$ if $\tau^i \le u$ and $x_u^i = x_0^i$ otherwise. Then (a) for every u, the law of $\mathcal{G}_u$ given $(\mathcal{G}_1, d)$ is Definition 1; (b) given x_1^i, entry i is a one-jump process from x_0^i to x_1^i at time τ^i; and (c) $\mathcal{G}_0$ is independent of $\mathcal{G}_1$.*

Proof. For each entry, $P(x_u^i = x \mid x_1^i, d) = P(\tau^i \le u)\,\delta_{x_1^i}(x) + P(\tau^i > u)\,\boldsymbol{q}_d^i(x) = u\,\delta_{x_1^i}(x) + (1-u)\,\boldsymbol{q}_d^i(x)$. The pairs (x_0^i, τ^i) are independent across entries given $(\mathcal{G}_1, d)$, so the entries are conditionally independent and their joint law is Equation 1 at time u. Parts (b) and (c) hold by construction. □

This coupling is the mixture path of Gat et al. (2024) with schedule $\kappa_u = u$ and source $\boldsymbol{q}_d$. Its source sample $\mathcal{G}_0$ is the state Algorithm 2 is initialized with.

Lemma A.2 (Conditional bridge: exact, bounded and domain-free)**.** *Let $0 \le t < s \le 1$ and $a = a(t,s)$.*

- *(a) $P(\mathcal{G}_s \mid \mathcal{G}_t, \mathcal{G}_1, d) = \prod_i B_{t\to s}(x_s^i \mid x_t^i, x_1^i)$.*
- *(b) $a \in (0,1]$, so every $B_{t\to s}(\cdot \mid x_t, x_1)$ is a probability vector; $a \to 0$ as $s \downarrow t$, and $a = 1$ if and only if $s = 1$.*
- *(c) $B_{t\to s}$ is the exact solution on $[t,s]$ of the coupling's conditional forward equation, whose generator is unbounded as $u \to 1$.*
- *(d) $B_{s\to r} \circ B_{t\to s} = B_{t\to r}$ for $t < s < r \le 1$.*
- *(e) The first-order (Euler) step of that equation from t, with step $s - t$, equals $B_{t\to s}$.*
- *(f) $B_{t\to s}$ depends only on (t, s, x_t, x_1); it does not depend on d, $\boldsymbol{q}_d$ or $\mathbb{V}_d$.*

Proof. (a) Fix an entry and drop the index i. If $x_t \ne x_1$, then $\tau > t$ and $x_0 = x_t$, so $x_s = x_1$ exactly when $\tau \le s$, which has probability $P(\tau \le s \mid \tau > t) = \frac{s-t}{1-t} = a$; otherwise $x_s = x_0 = x_t$. If $x_t = x_1$, then either $\tau \le t$, or $\tau > t$ and $x_0 = x_1$; in both cases $x_u = x_1 = x_t$ for every $u \ge t$, so $x_s = x_t$, which is $a\,\delta_{x_1} + (1-a)\,\delta_{x_t}$ because $\delta_{x_1} = \delta_{x_t}$. Given $\mathcal{G}_1$, the pairs (x_0^i, τ^i) are mutually independent, so the event that decides x_s^i is independent of the other entries' states. Hence $P(x_s^i \mid \mathcal{G}_t, \mathcal{G}_1, d) = B_{t\to s}(x_s^i \mid x_t^i, x_1^i)$, and the entries evolve independently, which gives the product.

(b) $0 < s - t \le 1 - t$ gives $a \in (0,1]$, so $B_{t\to s}$ is a convex combination of point masses.

(c) Given x_1, an unrevealed entry jumps to x_1 at the hazard rate of τ, $f_\tau(u)/P(\tau > u) = 1/(1-u)$. The conditional generator is therefore $R_u(y \to x \mid x_1) = \frac{\delta_{x_1}(x) - \delta_y(x)}{1-u}$, which vanishes for $y = x_1$ and is unbounded as $u \to 1$. The forward equation for $p_u = P(x_u = \cdot \mid x_t, x_1)$ on $[t,s]$ is

$$\partial_u p_u \;=\; \sum\nolimits_y p_u(y)\, R_u(y \to \cdot \mid x_1) \;=\; \frac{\delta_{x_1} - p_u}{1-u}, \qquad p_t = \delta_{x_t}. \tag{9}$$

With $r_u = p_u - \delta_{x_1}$ it reads $\partial_u r_u = -r_u/(1-u)$, so $r_s = r_t\, e^{-\Lambda(t,s)}$ with $\Lambda(t,s) = \int_t^s \frac{du}{1-u} = \log\frac{1-t}{1-s}$, and

$$p_s \;=\; \big(1 - e^{-\Lambda(t,s)}\big)\,\delta_{x_1} + e^{-\Lambda(t,s)}\,\delta_{x_t}, \qquad e^{-\Lambda(t,s)} \;=\; \frac{1-s}{1-t} \;=\; 1 - a. \tag{10}$$

The singularity of the generator enters the finite-step solution only through the survival factor $e^{-\Lambda(t,s)} \in [0,1)$. Every step is therefore bounded, and $e^{-\Lambda(t,s)} \to 0$ as $s \to 1$ forces full resolution at the endpoint.

(d) Because δ_{x_1} is absorbing, composing two kernels keeps the form $a\,\delta_{x_1} + (1-a)\,\delta_{x_t}$, now with $1 - a = (1 - a_{ts})(1 - a_{sr}) = \frac{1-s}{1-t} \cdot \frac{1-r}{1-s} = 1 - a_{tr}$. Given x_1, one step therefore equals any subdivision of the same interval: there is no discretization error.

(e) $\delta_{x_t} + (s-t)\, R_t(x_t \to \cdot \mid x_1) = \big(1 - \frac{s-t}{1-t}\big)\delta_{x_t} + \frac{s-t}{1-t}\,\delta_{x_1} = B_{t\to s}$, because $e^{-\Lambda(t,s)}$ is affine in s. The coincidence is specific to the linear schedule $\kappa_u = u$ (a general schedule gives $1 - a = \frac{1-\kappa_s}{1-\kappa_t}$). The grid ψ of Algorithm 2 only chooses which times are visited, so it is unaffected.

(f) is read off the formula. □

A worked example. Take one edge entry whose clean class is a single bond, observed at $t = 0.5$ and stepped to $s = 0.75$, so $a = 0.25/0.5 = 0.5$. In the reveal picture, a is the chance that a clock which has not rung by t rings in $(t, s]$: half of the remaining time is used up, so half of the still-unrevealed entries are revealed. If the entry currently shows "no edge", it becomes a single bond with probability 0.5 and otherwise stays; if it already shows a single bond, it stays, because either way it is correct. Splitting the same interval into $0.5 \to 0.625 \to 0.75$ gives $a = 0.25$ and then $a = \frac{1}{3}$; the probability of staying unrevealed across both is $0.75 \cdot \frac{2}{3} = 0.5$, the same as the single step. This is part (d): the step size does not matter.

Why this matters. Lemma A.2 gives the sampler three properties. It is *exact*: given the clean graph, one step of any size is the true transition, so the only error a sampler can make is in estimating the clean graph. It is *bounded*: although the underlying jump rate $1/(1-u)$ blows up near $u = 1$, the step only ever uses a probability $a \in (0, 1]$, so nothing has to be clipped. And it is *domain-free*: all domains share one parameter-free step, and the domain can only act through the prediction of the clean graph and the vocabulary it is restricted to. The final remark of this appendix contrasts this with rate-based samplers, where the first two properties fail.

From one clean graph to the true posterior. Lemma A.2 holds for a known clean graph, but a sampler only has a prediction of it. Lemma A.3 is the bridge between the two: if we average over where an entry might be at time t, the step lands exactly on its time-s marginal, and the prior merely rides along.

Lemma A.3 (Conditional transport)**.** *For every* x_1, $\sum_{x_t} P_t(x_t \mid x_1, d)\, B_{t\to s}(\cdot \mid x_t, x_1) = P_s(\cdot \mid x_1, d)$.

Proof. The left side is $\sum_{x_t} \big[t\,\delta_{x_1}(x_t) + (1-t)\,\boldsymbol{q}_d(x_t)\big]\big[a\,\delta_{x_1} + (1-a)\,\delta_{x_t}\big] = \big[a + (1-a)t\big]\,\delta_{x_1} + (1-a)(1-t)\,\boldsymbol{q}_d$, which equals $s\,\delta_{x_1} + (1-s)\,\boldsymbol{q}_d$ because $(1-a)(1-t) = 1-s$ and $a + (1-a)t = 1 - (1-a)(1-t) = s$. The prior appears in the input and output marginals, never in the kernel. □

Proof of Proposition 1. (i) is Lemma A.2(a) together with (f).

(ii) The strategy is to condition on the clean graph, where entries evolve independently and Lemma A.3 applies to each one, and then to average over the clean graph with Bayes' rule. By the law of total probability over the clean graph and (i),

$$P(\mathcal{G}_s \mid \mathcal{G}_t, d) \;=\; \sum\nolimits_{\mathcal{G}_1} \Pi_d(\mathcal{G}_1 \mid \mathcal{G}_t) \prod\nolimits_i B_{t\to s}(x_s^i \mid x_t^i, x_1^i). \tag{11}$$

Summing Equation 11 over every entry but i, each other factor sums to one, which leaves $P(x_s^i \mid \mathcal{G}_t, d) = \sum_x \pi_d^i(x \mid \mathcal{G}_t)\, B_{t\to s}(x_s^i \mid x_t^i, x) = a\,\pi_d^i(x_s^i \mid \mathcal{G}_t) + (1-a)\,\delta_{x_t^i}(x_s^i)$: this is Equation 2. For transport, Bayes' rule $P_t(\mathcal{G}_t \mid d)\,\Pi_d(\mathcal{G}_1 \mid \mathcal{G}_t) = P_{\text{data}}(\mathcal{G}_1 \mid d)\,P_t(\mathcal{G}_t \mid \mathcal{G}_1, d)$ and the entrywise factorization of $P_t(\mathcal{G}_t \mid \mathcal{G}_1, d)$ give

$$\begin{aligned}\sum\nolimits_{\mathcal{G}_t} P_t(\mathcal{G}_t \mid d)\, P(\mathcal{G}_s \mid \mathcal{G}_t, d) &= \sum\nolimits_{\mathcal{G}_1} P_{\text{data}}(\mathcal{G}_1 \mid d) \prod\nolimits_i \sum\nolimits_{x_t^i} P_t(x_t^i \mid x_1^i, d)\, B_{t\to s}(x_s^i \mid x_t^i, x_1^i) \\ &= \sum\nolimits_{\mathcal{G}_1} P_{\text{data}}(\mathcal{G}_1 \mid d) \prod\nolimits_i P_s(x_s^i \mid x_1^i, d) \;=\; P_s(\mathcal{G}_s \mid d), \end{aligned} \tag{12}$$

with Lemma A.3 in the second step. The step size never entered, so induction over any grid $0 = t_0 < \cdots < t_K = 1$ carries $P_0(\cdot \mid d) = \otimes_i \boldsymbol{q}_d^i$ to $P_1(\cdot \mid d) = P_{\text{data}}(\cdot \mid d)$.

(iii) A step can only keep an entry's current class or replace it with a draw from $\boldsymbol{\pi}^i$, so if both lie in $\mathbb{V}_d^i$ the entry never leaves it. Formally, by (A1), $\mathcal{G}_0 \in \prod_i \mathbb{V}_d^i$. If $x_t^i \in \mathbb{V}_d^i$, the law $a\,\boldsymbol{\pi}^i + (1-a)\,\delta_{x_t^i}$ is supported on $\operatorname{supp}\boldsymbol{\pi}^i \cup \{x_t^i\} \subseteq \mathbb{V}_d^i$. Induction over the steps completes the proof. □

Remark (The graph-level reading fails). Reading Equation 2 at the level of whole graphs, as $a\,\Pi_d(\cdot \mid \mathcal{G}_t) + (1-a)\,\delta_{\mathcal{G}_t}$, resolves every entry at once with probability a. That kernel maps $P_t(\cdot \mid d)$ to $a\,P_{\text{data}}(\cdot \mid d) + (1-a)\,P_t(\cdot \mid d)$, which is not $P_s(\cdot \mid d)$ in general. For instance, take two entries and data that never use a class g with $\boldsymbol{q}_d(g) > 0$: the state (g, g) has probability $(1-s)^2\,\boldsymbol{q}_d(g)^2$ under P_s but $(1-s)(1-t)\,\boldsymbol{q}_d(g)^2$ under the graph-level kernel. Entries must be resolved independently, as in Equation 11.

The graph-level reading is tempting because Equation 2 has the same "a times prediction plus $(1-a)$ times current state" shape. The difference is how many coins are flipped. In the forward process every entry has its own clock, so at any time some entries are revealed and others are not. The graph-level kernel flips one coin for the whole graph, which reveals all entries together or none, and therefore produces correlations between entries that the forward process never has. In the example, the two entries of (g, g) should each stay unrevealed through the step independently, which costs a factor $(1-a)$ per entry; the single shared coin charges that factor only once, so (g, g) is overweighted by $1/(1-a)$.

Does the noisy graph already reveal its domain? Proposition 1 leaves all domain dependence to the posterior $\boldsymbol{\pi}$. Conditioning the network on d is only useful to the extent that the noisy graph does not already reveal d, so we next locate where d enters the exact posterior and measure how much the state alone gives away.

Lemma A.4 (Where the domain enters $\boldsymbol{\pi}$). *(a)* $\Pi_d(\mathcal{G}_1 \mid \mathcal{G}_t) \propto P_{\text{data}}(\mathcal{G}_1 \mid d) \prod_i \big[t\,\delta_{x_1^i}(x_t^i) + (1-t)\,\boldsymbol{q}_d^i(x_t^i)\big]$*, so d enters only through the domain's data law and its prior* $\boldsymbol{q}_d = \mathbf{Q}_{\text{pr}}^\top \boldsymbol{e}_d$.

(b) Under (A2), $\operatorname{supp}\pi_d^i(\cdot \mid \mathcal{G}_t) \subseteq \mathbb{V}_d^i$.

(c) A model that ignores d can at best estimate the mixture $\Pi_\varnothing = \sum_{d'} P(d' \mid \mathcal{G}_t)\,\Pi_{d'}$*, where* $P(d' \mid \mathcal{G}_t) \propto P(d')\,P(n \mid d')\,P_t(\mathcal{G}_t \mid d', n)$*, and* $d_{\text{TV}}(\Pi_\varnothing, \Pi_d) \le 1 - P(d \mid \mathcal{G}_t)$.

(d) If some $x_t^i \notin \mathbb{V}_{d'}^i$*, then* $P(d' \mid \mathcal{G}_t) = 0$.

(e) For a single entry with clean marginal p_d*,* $\pi_d(\cdot \mid x_t) = \lambda_d\,\delta_{x_t} + (1-\lambda_d)\,p_d$ *with* $\lambda_d = \frac{t\,p_d(x_t)}{t\,p_d(x_t) + (1-t)\,\boldsymbol{q}_d(x_t)}$.

Proof. (a) is Bayes' rule with the likelihood Equation 1, and (e) is (a) for a single entry. (b) $\Pi_d(\mathcal{G}_1 \mid \mathcal{G}_t) > 0$ requires $P_{\text{data}}(\mathcal{G}_1 \mid d) > 0$. (c) The mixture form is Bayes' rule over d', with the node count part of the observed state. Then $\Pi_\varnothing - \Pi_d = \sum_{d' \neq d} P(d' \mid \mathcal{G}_t)\,(\Pi_{d'} - \Pi_d)$, and each difference has total variation at most one. (d) Under (A1) and (A2), $P_t(x_t^i \mid x_1^i, d') > 0$ only if $x_t^i \in \{x_1^i\} \cup \operatorname{supp}\boldsymbol{q}_{d'}^i \subseteq \mathbb{V}_{d'}^i$. □

In words, (c) says that a domain-blind model is wrong only in proportion to its uncertainty about which domain it is looking at: if the state leaves no doubt about d, the blind and conditional posteriors coincide. (d) says that a single class outside a domain's vocabulary removes that domain from consideration entirely. Table 4 measures how much doubt actually remains in our five domains.

Consequences. Because the priors are domain-specific and n is drawn from the domain's size distribution, the noisy state already identifies the domain almost surely (Table 4). Masks and priors separate structural from molecular domains exactly, by Lemma A.4(d) and Proposition 1(iii), and the only weakly separated case is an SBM graph with $n = 64$ against planar. Under a shared prior ($\boldsymbol{q}_d = \boldsymbol{q}$) the prior channel would carry no evidence. By (c), a domain-blind posterior is therefore close to the domain-conditional one; what the conditioning $\boldsymbol{c}$ adds is that the network does not have to infer d from these channels, and $\boldsymbol{c}$ is also the only channel that interpolation (Section 4.3) moves. By (e), when $\boldsymbol{q}_d$ equals the domain's class marginal, as with DiG^2's marginal prior up to size effects, $\lambda_d \equiv t$: one entry's own state carries no domain evidence, which therefore lives in the joint configuration of $\mathcal{G}_t$.

Table 4: How the noisy state itself separates the five domains at $t = 0$ (Lemma A.4(c,d)). Evidence is the expected log-likelihood ratio of the priors; sizes and priors are measured on the training split.

Domain pair	Separated by	Evidence
structural vs. molecular	vocabulary (exact)	disjoint node supports (one generic class vs. heavy atoms), and disjoint sizes ($n \geq 44$ vs. $n \leq 38$)
planar vs. tree	prior	both always have $n = 64$; edge densities 0.088 vs. 0.031 give 53–73 nats
SBM vs. planar, tree	node count, then prior	$P(n = 64 \mid \text{SBM}) = 0.031$; at $n = 64$ the prior gives 43–58 nats against tree but only 0.6 nats against planar (densities 0.081 vs. 0.088)
QM9 vs. ZINC	size, prior, partial vocabulary	sizes overlap only for $6 \leq n \leq 9$; 0.05 nats per node and 0.17 per pair; a ZINC state containing P, S, Cl, Br or I rules out QM9 exactly

Masking and guidance. Masking sets the logits of forbidden classes to a large negative number, which is the same as renormalizing the prediction onto the allowed classes. Lemma A.5 shows that this is exact in floating point, that it can never move a prediction further from a target that lives on the allowed classes, and that it survives guidance only if both branches are masked.

Lemma A.5 (Vocabulary masking)**.** *Let $\boldsymbol{\ell} \in \mathbb{R}^C$, let $\boldsymbol{\ell}\restriction_{\mathbb{V}}$ set the logits of classes outside $\mathbb{V}$ to $-M$, and let $\boldsymbol{\pi} = \mathrm{softmax}(\boldsymbol{\ell})$.*

(a) As $M \to \infty$, $\mathrm{softmax}(\boldsymbol{\ell}\restriction_{\mathbb{V}}) \to \boldsymbol{\pi}_{\mathbb{V}}$, the renormalized restriction $\boldsymbol{\pi}_{\mathbb{V}}(x) = \boldsymbol{\pi}(x)\,\mathbf{1}[x \in \mathbb{V}]/\boldsymbol{\pi}(\mathbb{V})$; with $M = 10^9$ the off-$\mathbb{V}$ probabilities are exactly zero in single precision.

(b) For every $\boldsymbol{\pi}^\star$ supported on $\mathbb{V}$, $d_{\mathrm{TV}}(\boldsymbol{\pi}_{\mathbb{V}}, \boldsymbol{\pi}^\star) \leq d_{\mathrm{TV}}(\boldsymbol{\pi}, \boldsymbol{\pi}^\star)$.

(c) If both guidance branches are masked, the guided logit of every class outside $\mathbb{V}$ is $-M + g\,(-M - (-M)) = -M$ for every $g \in \mathbb{R}$. If only $\boldsymbol{\ell}^c$ is masked it is $(1-g)\,\ell^u - gM$, which fails for $g \leq 0$; if only $\boldsymbol{\ell}^u$ is masked it is $(g-1)M + g\,\ell^c$, which removes the mask at $g = 1$ and inverts it for $g > 1$.

Proof. (a) The masked logits leave the relative weights inside $\mathbb{V}$ unchanged, and e^{-M} vanishes. (b) For distributions $\boldsymbol{p}, \boldsymbol{r}$, $d_{\mathrm{TV}}(\boldsymbol{p}, \boldsymbol{r}) = \sum_x (\boldsymbol{r}(x) - \boldsymbol{p}(x))^+$. Taking $\boldsymbol{r} = \boldsymbol{\pi}^\star$, the terms outside $\mathbb{V}$ vanish for both arguments, and $\boldsymbol{\pi}_{\mathbb{V}} \geq \boldsymbol{\pi}$ on $\mathbb{V}$ makes every remaining term no larger. (c) is direct substitution into Equation 6. □

By (b) and Lemma A.4(b), masking can only reduce the estimator's error against the exact posterior. By (c), premise (A3) is necessary for the guided sampler of Equation 6 at the guidance weights we use, $g(d) \geq 2$. To see the failure concretely, take $g = 2$ and mask only the unconditional branch: a forbidden class receives the guided logit $M + 2\,\ell^c$, so the mask, instead of removing the class, makes it almost certain to be drawn. Guidance subtracts the unconditional logits, and subtracting a one-sided $-M$ turns it into $+M$.

Corollary A.6 (Guided sampler)**.** *Let $\boldsymbol{\pi}^i_g \propto (\boldsymbol{\pi}^{u,i})^{1-g}(\boldsymbol{\pi}^{c,i})^g$ on $\mathbb{V}^i_d$, the entrywise form of Equation 6 under (A3).*

(a) $\boldsymbol{\pi}^i_g$ is a probability vector on $\mathbb{V}^i_d$ for every $g \in \mathbb{R}$, so the kernel $a\,\boldsymbol{\pi}^i_g + (1-a)\,\delta_{x^i_t}$ is a valid law for every (t, s, g) and Proposition 1(iii) applies.

(b) If $\boldsymbol{\pi}^{c,i} = \pi^i_d$ and $\boldsymbol{\pi}^{u,i} = \pi^i_\varnothing$, the entry marginal of $\Pi_\varnothing$, then $\boldsymbol{\pi}^i_g(x) \propto \pi^i_\varnothing(x \mid \mathcal{G}_t)\,P(d \mid x^i_1 = x, \mathcal{G}_t)^g$.

(c) For $g \neq 1$ the sampler no longer targets $P_{\mathrm{data}}(\cdot \mid d)$, so Proposition 1(ii) is claimed for $g = 1$ only; Algorithm 2 uses $g = 1$ at its final step, where $a = 1$.

Proof. (a) follows from Lemma A.5(c): the guided logits are finite on $\mathbb{V}^i_d$ and equal to $-M$ off it. (b) By Bayes' rule, $P(d \mid x^i_1 = x, \mathcal{G}_t) = P(d \mid \mathcal{G}_t)\,\pi^i_d(x \mid \mathcal{G}_t)/\pi^i_\varnothing(x \mid \mathcal{G}_t)$, so $(\pi^i_\varnothing)^{1-g}(\pi^i_d)^g = \pi^i_\varnothing\,(\pi^i_d/\pi^i_\varnothing)^g \propto \pi^i_\varnothing\,P(d \mid x^i_1 = x, \mathcal{G}_t)^g$. (c) At $g = 1$, $\boldsymbol{\pi}_g = \boldsymbol{\pi}^c$. □

Guidance therefore tempers the per-entry domain posterior, pushing each entry toward the classes that are most indicative of d. Read (b) as a reweighting: start from the domain-blind prediction and multiply each class by how strongly it points to d, raised to the power g. At $g = 1$ this recovers plain conditional sampling; at $g > 1$ it sharpens toward classes that are characteristic of d, at the price of no longer sampling $P_{\mathrm{data}}(\cdot \mid d)$ exactly (part (c)). For $g > 0$ this tilt already vanishes outside $\mathbb{V}_d$, because $P(d \mid x_1^i = x, \mathcal{G}_t) = 0$ there; the mask enforces on the estimator a property the exact tilt has. Applying guidance to transition rates instead, as $R_u^{1-g} R_c^g$, is unbounded wherever $R_u \to 0$ faster than R_c and must be clipped; the logit-level form Equation 6 is bounded by construction.

What the network's error costs. Everything so far assumed the exact posterior Π_d. In practice a network predicts one independent distribution per entry, and the sampler takes T steps with it. Proposition A.7 bounds the resulting error in the generated distribution. The idea of the proof is a telescoping sum: each step adds the error of its own prediction, and errors made at earlier steps cannot grow afterwards, because applying any Markov kernel to two distributions never moves them further apart in total variation.

Proposition A.7 (Bounded error of the implemented sampler)**.** *Fix a grid $0 = t_0 < \cdots < t_T = 1$ with $a_k = a(t_k, t_{k+1})$. At step k, let $\hat{\Pi}_k(\cdot \mid \mathcal{G})$ be any law over clean graphs and let $\hat{K}_k(\cdot \mid \mathcal{G}) = \sum_{\mathcal{G}_1} \hat{\Pi}_k(\mathcal{G}_1 \mid \mathcal{G}) \prod_i B_{t_k \to t_{k+1}}(\cdot \mid x^i, x_1^i)$; the network's factorized prediction $\hat{\Pi}_k = \otimes_i \hat{\pi}_k^i$ gives the step of Algorithm 2, $\prod_i [a_k\, \hat{\pi}_k^i + (1 - a_k)\, \delta_{x^i}]$. Let $\hat{P}_1$ be the law of the last iterate started from $\mathcal{G}_0 \sim \otimes_i \boldsymbol{q}_d^i$, and let $\Pi_{d,S}$ denote the marginal of Π_d on a set of entries S. Then*

$$d_{\mathrm{TV}}\big(\hat{P}_1, P_{\mathrm{data}}(\cdot \mid d)\big) \le \sum_{k=0}^{T-1} \mathbb{E}_{\mathcal{G} \sim P_{t_k}(\cdot \mid d)}\, \rho_k(\mathcal{G}), \tag{13}$$

$$\rho_k(\mathcal{G}) \le \mathbb{E}_S\, d_{\mathrm{TV}}\big(\hat{\Pi}_{k,S}, \Pi_{d,S}\big) \le \big(1 - (1 - a_k)^{N_n}\big)\, d_{\mathrm{TV}}\big(\hat{\Pi}_k, \Pi_d\big),$$

where $\rho_k(\mathcal{G}) = d_{\mathrm{TV}}(\hat{K}_k(\cdot \mid \mathcal{G}), K_k(\cdot \mid \mathcal{G}))$, K_k is the exact kernel Equation 11, and S includes each entry independently with probability a_k. For the factorized prediction, $d_{\mathrm{TV}}(\hat{\Pi}_{k,S}, \Pi_{d,S}) \le \sum_{i \in S} d_{\mathrm{TV}}(\hat{\pi}_k^i, \pi_d^i) + d_{\mathrm{TV}}(\otimes_{i \in S} \pi_d^i, \Pi_{d,S})$, and the last term vanishes when $|S| \le 1$.

Proof. Write $P_k = P_{t_k}(\cdot \mid d)$ and $\hat{P}_k$ for the law of the k-th iterate, so that $P_{k+1} = P_k K_k$ by Proposition 1(ii) and $\hat{P}_{k+1} = \hat{P}_k \hat{K}_k$. Then $\hat{P}_{k+1} - P_{k+1} = (\hat{P}_k - P_k)\, \hat{K}_k + P_k\, (\hat{K}_k - K_k)$. A Markov kernel does not increase total variation, so $d_{\mathrm{TV}}(\hat{P}_{k+1}, P_{k+1}) \le d_{\mathrm{TV}}(\hat{P}_k, P_k) + \mathbb{E}_{\mathcal{G} \sim P_k} \rho_k(\mathcal{G})$; summing from $\hat{P}_0 = P_0$ gives the first inequality. For the second, the product $\prod_i [a_k\, \delta_{x_1^i} + (1 - a_k)\, \delta_{x^i}]$ is the law of "draw S, copy x_1^i on S and keep x^i elsewhere". Both kernels are therefore mixtures over the same law of S, of the S-marginal of $\hat{\Pi}_k$ or Π_d, and joint convexity of total variation bounds ρ_k by $\mathbb{E}_S d_{\mathrm{TV}}(\hat{\Pi}_{k,S}, \Pi_{d,S})$. The term $S = \emptyset$ vanishes and marginalization does not increase total variation, which gives the third inequality. The last claim is the triangle inequality with subadditivity of total variation over product measures; for $|S| \le 1$ the product of marginals equals the marginal. □

Four consequences follow.

- *No discretization error.* With the exact posterior every term vanishes for any T.
- *Errors add rather than compound.* Each step contributes at most the posterior error on the entries it resolves, weighted by $a_k \le 1$. On the grid $t_k = \psi(k/T)$ with $\psi(u) = 2u - u^2$, $1 - a_k = \left(\frac{m-1}{m}\right)^2$ with $m = T - k$, so $a_0 = \frac{2T-1}{T^2} \approx 0.01$ at $T = 200$ and $a_{T-1} = 1$: early steps resolve few entries and so expose little of the network's error.
- *Masking helps.* It can only shrink the first term of the factorized bound (Lemma A.5(b)).
- *Guidance is a deliberate deviation.* With guidance, the same bound measures the deviation from $P_{\mathrm{data}}(\cdot \mid d)$ that the tilt introduces on purpose.

We do not state a numerical bound, because the entrywise sum grows with N_n; the value of the result is its structure.

Remark (Relation to chain- and rate-based samplers). Chain-based models (D3PM, DiGress) also compute their reverse step in closed form, from a fixed forward transition matrix and a predicted clean graph; neither learns a transition object. Their forward chain resamples the noise at every step, however, so their posterior spreads mass over other noise classes, whereas $B_{t\to s}$ is supported on $\{x_t, x_1\}$. Rate-based samplers such as DeFoG integrate a continuous-time generator built from the predicted clean graph with first-order steps, which must be clipped near $t = 1$, where the rates grow as $1/(1-t)$. Equation 2 is the exact integration of the mixture-path generator (Lemma A.2(c,e)), which is why it needs neither.

B Limitations

The current benchmarking study is foundational, establishing that multi-domain graph generation is possible. Therefore, the vocabulary formulation is intentionally restricted to 11 available node-classes and 5 edge-classes. Future work with graph generative models could involve exciting applications learning with larger domain vocabularies.

C Implementation Details

Conditioning. Every conditioning signal the denoiser receives is assembled into a single vector $\boldsymbol{c}$ from two sources. A learned, orthogonally-initialized domain embedding table, followed by a residual MLP, supplies $\boldsymbol{z}_d$, with one extra row reserved for the classifier-free-guidance null token so that dropped-out graphs receive a dedicated learned null vector $\boldsymbol{z}_\varnothing$ rather than a zero or uniform one; a small GNN structural encoder $f_{\mathrm{struct}}(\cdot\,;\boldsymbol{\omega})$ summarizes the current noisy state and is fused with an exponential-moving-average estimate of that domain's clean-graph statistics,

$$\boldsymbol{s}_{\mathrm{cur}} = f_{\mathrm{struct}}(\boldsymbol{X}_t, \mathbf{E}_t\,;\boldsymbol{\omega}), \qquad \boldsymbol{s} = f_{\mathrm{fuse}}\big([\boldsymbol{s}_{\mathrm{proto},d} \| \boldsymbol{s}_{\mathrm{cur}}]\big), \tag{14}$$

$$\boldsymbol{s}_{\mathrm{proto},d} \leftarrow \lambda\, \boldsymbol{s}_{\mathrm{proto},d} + (1-\lambda)\, \frac{\boldsymbol{m}_d - \boldsymbol{\mu}}{\|\boldsymbol{m}_d - \boldsymbol{\mu}\|}, \qquad \boldsymbol{m}_d = \frac{1}{|\mathcal{B}_d|} \sum_{\mathcal{G}_1 \in \mathcal{B}_d} f_{\mathrm{struct}}(\boldsymbol{X}_1, \mathbf{E}_1\,;\boldsymbol{\omega}), \tag{15}$$

where $\mathcal{B}_d$ holds the clean training graphs of domain d in the current batch and $\boldsymbol{\mu} \leftarrow \beta_\mu \boldsymbol{\mu} + (1-\beta_\mu)\,\mathrm{mean}_{d:\,\mathcal{B}_d \neq \emptyset}\, \boldsymbol{m}_d$ is a running center shared by all domains ($\lambda = \beta_\mu = 0.99$). Only domains present in the batch are written, and only in training steps; Algorithm 3 gives the update. Centering removes the component common to every domain, so the prototypes differ in direction rather than by a shared offset, and the ℓ_2 normalization keeps each written mean on the unit sphere. Of the two fused inputs, only $\boldsymbol{s}_{\mathrm{cur}}$ receives gradient from the denoising loss; the prototype is a detached running estimate over clean graphs. The two are concatenated, $\boldsymbol{c} = [\boldsymbol{z}_d \| \boldsymbol{s}]$. The unconditional branch of guidance replaces the two domain-indexed terms and keeps the third,

$$\boldsymbol{c}_\varnothing = \big[\, \boldsymbol{z}_\varnothing \,\|\, f_{\mathrm{fuse}}([\mathbf{0} \| \boldsymbol{s}_{\mathrm{cur}}]) \,\big], \tag{16}$$

so $\boldsymbol{s}_{\mathrm{cur}}$ still reads the noisy graph in both branches. The contrast $\boldsymbol{\ell}^c - \boldsymbol{\ell}^u$ in Equation 6 is therefore between domain identities *given the same observed structure*, not between a structure-aware and a structure-blind prediction. During training each graph is independently switched from $\boldsymbol{c}$ to $\boldsymbol{c}_\varnothing$ with probability p_{drop}.

Domain-conditioned attention. Each of transformer layers injects $\boldsymbol{c}$ through three near-identity-at-initialization adapters: a residual added to each node's query/key/value projection, a multiplicative gate on the attention logits, and a FiLM affine transform applied to the node, edge, and global outputs. The head gate acts on the per-feature score tensor before the layer rewrites the edges from it, so it modulates the new edge features as well as the attention weights. All three are scaled by $\tanh$ of learned gains initialised small (Table 5), so the layer starts close to an unconditioned transformer and deviates from it only as training progresses. Outside the stack, $\boldsymbol{c}$ enters at three more places. It is concatenated into the global input before the input MLP, so $\boldsymbol{y}$ carries it from the first layer. The timestep embedding is FiLM-modulated by the same conditioning vector $\boldsymbol{c}$, with both a scale and a shift at a fixed gain, $\boldsymbol{\tau} \leftarrow \boldsymbol{\tau} \odot (1 + 0.25 \tanh \boldsymbol{\gamma}) + 0.25 \tanh \boldsymbol{\beta}$, where $(\boldsymbol{\gamma}, \boldsymbol{\beta})$ come from a two-layer MLP on $\boldsymbol{c}$ — that is, from $\boldsymbol{z}_d$ and the structural summary $\boldsymbol{s}$ together, not from the domain embedding alone — giving the network a domain-aware notion of "how far along" denoising is; it is applied once before the stack rather than per layer. Finally, the multi-depth readout weights $w(\boldsymbol{c}) = \mathrm{softmax}(\mathrm{MLP}(\boldsymbol{c}))$ are computed from $\boldsymbol{c}$ with a zero-initialized last layer, so they start as a uniform average over $\mathbb{K}$.

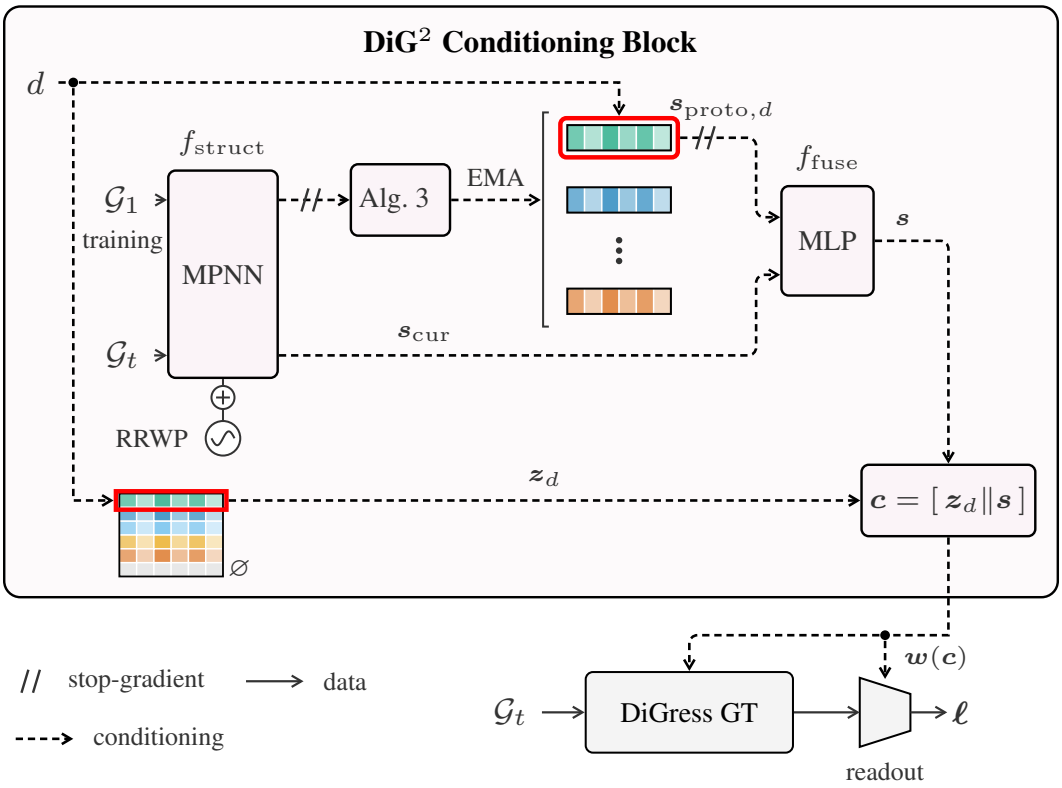


Figure 5: **The DiG² conditioning block.** $c = [z_d \| s]$ joins a learned domain embedding z_d with a structural summary s, the highlighted table row and prototype are those of domain d, and the shaded card is the DiG² conditioning block. Everything outside the card is the inherited transformer and readout.

Deep supervision. Non-terminal readout depths ($\mathbb{K} \setminus \{L\}$) each receive their own cross-entropy term against the same target, restricted to $\mathbb{V}_d$ exactly as the terminal loss is, but with unit node and edge weights ($\text{CE}_X + \text{CE}_E$) rather than the terminal loss's $(\lambda_X, \lambda_E) = (1, 5)$ (Table 5). The terms are uniformly averaged and scaled by $\beta = 1.0$; this is the $\mathcal{L}_{\text{deep-sup}}$ term in Algorithm 1.

Vocabulary restriction. $\mathbb{V}_d$ is the semantically declared class support for domain d. The restriction operator $\restriction_{\mathbb{V}_d}$ used throughout is a hard mask applied to logits before any softmax — training targets and both classifier-free-guidance branches at every sampling step alike — implemented by setting every logit of a class outside $\mathbb{V}_d$ to $-M$ with $M = 10^9$, so those classes receive exactly zero probability. Masking both guidance branches to the same $\mathbb{V}_d$ keeps the cond/uncond difference in Equation 6 purely semantic, and it is necessary for the mask to survive guidance at $g(d) > 1$ (Lemma A.5).

Algorithm 3 Prototype write (once per training step, after the loss; no gradient)

Require: clean graphs of the batch, grouped by true domain into $\mathcal{B}_d$;
prototypes $s_{\text{proto},d}$ (init. **0**); center $\boldsymbol{\mu}$; momenta $\lambda = \beta_\mu = 0.99$
Ensure: $s_{\text{proto},d}$ updated for the domains in the batch only

$\boldsymbol{m}_d \leftarrow \text{mean}_{\mathcal{G}_1 \in \mathcal{B}_d} f_{\text{struct}}(\mathcal{G}_1\,; \boldsymbol{\omega})$ for each domain d in the batch ▷ domain mean
$\bar{\boldsymbol{m}} \leftarrow$ mean of these $\boldsymbol{m}_d$ ▷ over the domains in the batch
$\boldsymbol{\mu} \leftarrow \bar{\boldsymbol{m}}$ on the first write, else $\boldsymbol{\mu} \leftarrow \beta_\mu\, \boldsymbol{\mu} + (1 - \beta_\mu)\, \bar{\boldsymbol{m}}$ ▷ shared center
for each domain d in the batch **do**
    $s_{\text{proto},d} \leftarrow \lambda\, s_{\text{proto},d} + (1 - \lambda)\, (\boldsymbol{m}_d - \boldsymbol{\mu}) / \|\boldsymbol{m}_d - \boldsymbol{\mu}\|$ ▷ EMA toward centered unit mean
end for

Guidance schedule. $g(d)$ in Equation 6 is a per-domain lookup table with a shared fallback (Table 5); classifier-free-guidance dropout uses $p_{\text{drop}} = 0.05$.

Time discretization. Both training-time and sampling-time schedules reparameterize a uniform variable through $\psi(u) = 2u - u^2$, which places more steps near $t = 1$. At training time, $t = \psi(u)$ for $u \sim \text{Unif}(0, 1)$; at sampling time, Algorithm 2's $T = 200$ steps visit the grid $t_k = \psi(k/T)$. Proposition 1's resolve probability for the step from t_k to t_{k+1} is then $a_k = 1 - \left(1 - \frac{1}{T-k}\right)^2$, rising from $a_0 = \frac{2T-1}{T^2} \approx 0.01$ to $a_{T-1} = 1$.

Quantity	Value
Transformer depth L	12
Readout depths $\mathbb{K}$	$\{8, 10, 12\}$
Parameters	18M
Hidden dims (node, edge, global, heads)	256, 64, 128, 8
Feed-forward dims (node, edge, global)	256, 128, 2048
Domain-emb. dim / structural-encoder depth	32 / 8
Structural aux. features	RRWP, $k = 20$ Ma et al. (2023)
Prototype transform	centered, ℓ_2-normalized; momentum 0.99
Corruption prior	domain-specific
CE weights $(\lambda_X, \lambda_E, \lambda_y)$	$(1, 5, 0)$
Deep-supervision weight β (terminal excluded)	1.0
$g(d)$	SBM 3.0, others 2.0
Guidance dropout p_{drop}	0.05
Conditioning sites for $\boldsymbol{c} = [\boldsymbol{z}_d \| \boldsymbol{s}] \in \mathbb{R}^{64}$ (Figure 5, §C)	
Residual on $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$, per layer	$\boldsymbol{Q} \mathrel{+}= \tanh(\gamma_q)\,\text{cond}_q(\boldsymbol{c})$; γ init. 0.05
Attention-head gate, per layer	$\boldsymbol{Y} \mathrel{*}= 1 + \tanh(\gamma_h)\,\text{gate}(\boldsymbol{c})$; γ_h init. 0.05
FiLM on $\boldsymbol{X}, \boldsymbol{\mathsf{E}}, \boldsymbol{y}$, per layer	$u \mapsto u(1+\tanh(\gamma_f)a) + \tanh(\gamma_f)b$; γ_f init. 0.30
Timestep FiLM, once before the stack	scale and shift, fixed gain 0.25
Global input, once before the stack	concatenated before the input MLP
Readout gate $w(\boldsymbol{c})$ over $\mathbb{K}$	zero-init. last layer (uniform at init.)
Unconditional branch $\boldsymbol{c}_\varnothing$	learned null row; prototype zeroed; $\boldsymbol{s}_{\text{cur}}$ kept
Sample steps T / time distortion	200 / `polydec`
Node budget / steps per epoch / epochs	256 nodes / 250 / 10,000 (§C.1)
Molecular hydrogen convention	`remove_h = true` (heavy-atom only)
$\mathbb{D}$	planar, sbm, tree, qm9, zinc

Table 5: DiG^2 Hyperparameters.

A note on $g(d)$. The per-domain guidance table used by the reported runs is only sparsely calibrated, SBM at 3.0 and every other domain at the shared fallback of 2.0.

Anchored fine-tuning. After base training, all benchmarked models are fine-tuned to improve domain metrics without affecting other domains ($\mathbb{D}_{\text{prot}}$) (Alg. 4). In our study, only the edge pathway train: attention projections, edge MLPs and edge readout heads, $\boldsymbol{\theta}_E$, 3.98M of the 18M parameters (22%). Every other weight is frozen, including the readout gate $\boldsymbol{\phi}$, the structural encoder $\boldsymbol{\omega}$, the embedding table and the domain prototypes, which are not written during fine-tuning. Because $\mathbb{D}_{\text{prot}}$ shares $\boldsymbol{\theta}_E$, its outputs would drift, so a frozen copy of the starting network, $F_{\boldsymbol{\theta}_0}$, acts as an anchor: on graphs from $\mathbb{D}_{\text{prot}}$ the student is penalized by $\text{KL}(F_{\boldsymbol{\theta}_0} \,\|\, F_{\boldsymbol{\theta}})$, computed on the same noisy input and conditioning and with both predictions restricted to $\mathbb{V}_d$, averaged over nodes (KL_X) and over off-diagonal node pairs (KL_E) with $\lambda_E = 5$, matching the edge weight of the training loss. The task loss (cross-entropy and deep supervision) is applied to $\mathbb{D}_{\text{tgt}}$ only. Each loss term is averaged over the entries of its own graphs in the batch, so the balance between the two terms does not depend on the batch's mix of domains; a batch with no target graph takes an anchor-only step. P_{tune} draws domains per graph with weights 0.25 (SBM, PLANAR) and $1/6$ (each protected domain). We use $\lambda_{\text{anc}} = 12$, AdamW (AMSGrad) at learning rate $2{\times}10^{-5}$ ($10\times$ below base training) with a 200-step linear warmup, no weight decay and gradient clipping at 1.0, for $J = 25{,}000$ steps, and report the final checkpoint.

Algorithm 4 Anchored fine-tuning
(shaded lines differ from Alg. 1; no prototype writes)

Require: $\boldsymbol{\theta}_0$ (Alg. 1); edge pathway $\boldsymbol{\theta}_E \subset \boldsymbol{\theta}$; $\mathbb{D}_{\mathrm{tgt}}, \mathbb{D}_{\mathrm{prot}}$; $\lambda_{\mathrm{anc}}, \lambda_E$; J

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_0$; freeze a copy $F_{\boldsymbol{\theta}_0}$; train only $\boldsymbol{\theta}_E$
for $j = 1, \ldots, J$ **do**
  draw a batch $B \sim P_{\mathrm{tune}}$ ▷ reweighted
  for each $(\mathcal{G}_1, d) \in B$ **do**
    $u \sim \mathrm{Unif}(0,1)$; $t \leftarrow \psi(u)$
    $\mathcal{G}_t \sim P_t(\cdot \mid \mathcal{G}_1, d)$ ▷ noise to $\boldsymbol{q}_d$
    $\boldsymbol{s} \leftarrow f_{\mathrm{fuse}}([\boldsymbol{s}_{\mathrm{proto},d} \| f_{\mathrm{struct}}(\mathcal{G}_t)])$
    $\boldsymbol{c} \leftarrow [\boldsymbol{z}_d \| \boldsymbol{s}]$, or $\boldsymbol{c}_\varnothing$ w.p. p_{drop}
    $\hat{\mathcal{G}}_1 \leftarrow F_{\boldsymbol{\theta}}(\mathcal{G}_t, t, \boldsymbol{c}) \restriction_{\mathbb{V}_d}$
    $\hat{\mathcal{G}}_1^0 \leftarrow F_{\boldsymbol{\theta}_0}(\mathcal{G}_t, t, \boldsymbol{c}) \restriction_{\mathbb{V}_d}$ ▷ no gradient
  end for
  $\mathcal{L}_{\mathrm{task}} \leftarrow \mathrm{CE}(\hat{\mathcal{G}}_1, \mathcal{G}_1) + \mathcal{L}_{\text{deep-sup}}$ ▷ over $\mathbb{D}_{\mathrm{tgt}}$
  $\mathcal{L}_{\mathrm{anc}} \leftarrow \mathrm{KL}_X + \lambda_E \mathrm{KL}_E$ of $(\hat{\mathcal{G}}_1^0 \| \hat{\mathcal{G}}_1)$ ▷ over $\mathbb{D}_{\mathrm{prot}}$
  $\boldsymbol{\theta}_E \leftarrow$ AdamW step on $\nabla(\mathcal{L}_{\mathrm{task}} + \lambda_{\mathrm{anc}} \mathcal{L}_{\mathrm{anc}})$
end for
return $\boldsymbol{\theta}$

C.1 Data Loading and Node Budgets

Mixed-domain training batches. Batches are filled to a node budget rather than a graph count, so a single batch can hold graphs from every domain and size range. Each draw first picks a domain uniformly over the five domain names, then a training graph uniformly within that domain. The corpus is highly unbalanced (128 training graphs each for PLANAR, SBM and TREE, 97,734 for QM9 and 224,568 for ZINC), so this gives every domain the same draw probability whatever its size. Graphs are appended while the batch's total node count stays within 256. The first draw that would exceed the budget closes the batch and is discarded, not carried over. An epoch is 250 such batches, with a fresh sampler seed each epoch. Replaying the sampler over the training set (10,000 batches) gives 4.6 graphs and 217 nodes per batch on average; 96.8% of batches contain at least two domains (3.1 on average) and 0.8% contain a single graph. Two properties follow from the rule. First, the budget rejects large graphs more often, so SBM graphs above 128 nodes are 29.7% of the SBM pool but 22.8% of the SBM graphs consumed. Second, equal draw probability does not mean equal node exposure: SBM, TREE and PLANAR supply 33%, 26% and 25% of a batch's nodes, ZINC 11% and QM9 4%. The anchored fine-tuning (Algorithm 4) instead draws domains with fixed weights and carries an overflowing graph into the next batch rather than discarding it.

Independent structure within a batch. Each graph stays its own connected component. Collation offsets every graph's edge indices, so no edge crosses graphs. The batch is then made dense: node features are padded to the batch's largest graph, $N_{\max}$ (67 nodes at the median batch, 174 at the 99th percentile), and each graph gets its own $N_{\max} \times N_{\max}$ edge block, whose padded rows and columns are removed by a per-graph node mask. Padded cells and the diagonal carry no target class and drop out of the cross-entropy. Per-graph metadata travels with the batch. The domain index selects each graph's vocabulary mask $\mathbb{V}_d$. Classifier-free-guidance dropout is drawn per graph. The same index routes the graph's clean embedding into its domain's prototype (Algorithm 3). All domains share one vocabulary of 11 node and 5 edge classes. Each domain's local classes are mapped into it once, when the dataset is built: the structural domains use one generic node class and one edge class, and the molecular domains use atom types and bond orders. $\mathbb{V}_d$ records which global classes domain d can use. Padding has a compute cost: on average 62% of a batch's node slots and 52% of its edge slots belong to a real graph.

Sampling batches. Generation batches are domain-pure. Each batch generates graphs of a single domain under that domain's conditioning and holds $\lfloor 2048/\bar{n}_d \rfloor$ graphs, where $\bar{n}_d$ is the domain's mean training size: 19 graphs per batch for SBM, 32 for PLANAR and TREE, 89 for ZINC and 227 for QM9. Each graph's node count is drawn from its domain's empirical training-size histogram. Graphs are padded to the batch's largest node count and masked as in training, so each is denoised independently.

Memory and Time Computation. Within Table 6 of this section, we detail the peak memory usage and time required per-epoch of all benchmarked models during training and sampling for the benchmarking study. Within Table 7, we detail the dataset statistics for reproducibility. All tests were conducted on a 1 TB RAM machine with 48GB A6000 GPUs. All models were trained for 2.5M steps and fine-tuned for 25K steps, and given a 10-day training budget before stopping.

		Training		Sampling	
Method	Params (M)	s/epoch	Peak GB	s/graph	Peak GB
DiG²	18	87.2	18.7	5.22	5.8
G2PT	89	167.4	4.6	1.52	4.5
DeFoG†	18	53.9	14.0	4.35	2.8
DiGress†	18	27.7	16.6	4.71	2.8
HOG-Diff†	18	110.2	16.2	7.79	2.9
GraphBSI†	18	46.6	14.0	2.43	2.8
LGGM	18	26.7	17.1	3.78	3.6
Pard†	18	167.0	36.0	3.74	2.9
GraphRNN†	18	20.0	9.4	2.1	1.9
LGD†	27	127.9	19.1	4.7	2.0
SPECTRE†	18	489.5	14.9	3.0	5.4
GDSS†	18	188.8	15.7	3.44	2.5

Table 6: Hyperparameters, per-epoch timing and memory required for training and sampling

Dataset	Min Nodes	Max Nodes	Graphs Sampled
Planar	64	64	40
SBM	44	187	40
Tree	64	64	40
QM9	2	9	10000
ZINC	8	27	10000

Table 7: Dataset Statistics and Number of Graphs Sampled.

D BASELINE CONFIGURATIONS

Every arm in Table 1 is trained on the same corpus, splits, batching rule and evaluation harness, so the tables below separate what is *shared* from what is *architecture* and what is *specific to each generative process*. The training protocol is shared: AdamW (learning rate 2×10^{-4}, weight decay 10^{-12}), gradient-norm clipping at 1.0, a budget of 256 nodes per step, 250 steps per epoch for $10{,}000$ epochs, no weight EMA and seed 42. The exceptions are the upstream methods' own requirements or memory limits: LGD's denoiser uses learning rate 10^{-4} and its autoencoder trains for 500 epochs; SPECTRE uses separate generator and discriminator Adam optimizers (learning rate 10^{-4}, $(\beta_1, \beta_2) = (0.5, 0.9)$), weight EMA 0.995 and a 192-node budget; Pard uses at least 2 graphs per step. Table 8 gives each arm's neural backbone and its size relative to DiG^2, and Table 9 lists the parameters that belong to each arm's generative process, including how the domain signal enters it. Every configuration is sized on the corpus-wide vocabulary of 11 node and 5 edge classes and a maximum graph size of 187 nodes.

Model	Backbone (components, M)	Layers	Hidden	Heads	FFN	Params (M)
Graph transformers with node / edge / global streams (widths listed $X/E/y$)						
DiG^2	graph transformer	12	256/64/128	8	256/128/2048	18
DeFoG†	graph transformer	12	256/64/128	8	256/128/2048	18
GraphBSI†	graph transformer	12	256/64/128	8	256/128/2048	18
DiGress†	graph transformer	12	256/64/128	8	256/128/2048	18
LGGM	graph transformer	12	256/64/128	8	256/128/2048	18
Other score and diffusion networks (two-network arms listed first / second)						
HOG-Diff†	GNN + attention score net per phase (7.74 / 10.29)	5 / 5	320	8	—	18
GDSS†	score nets s_X, GMH (6.93) / s_A, GCN (11.37)	3 / 8	256	8	—	18
Pard†	PPGN diffusion (18.34) / PPGN block-size predictor (3.46)	8 / 6	512 / 256	8	—	22
LGD†	frozen graph-transformer autoencoder (1.05) / latent denoiser (25.88)	5 / 8	96 / 256	4 / 16	—	27
Autoregressive and adversarial						
GraphRNN†	graph-level GRU (16.24) / edge-level GRU (1.98)^e	5 / 5	736 / 256	—	—	18
G2PT	GPT decoder	12	768	12	3072	89
SPECTRE†	λ, SON, PPGN generators (10.16) / discriminators (7.72)	8	216	—	—	18

Table 8: Backbone architecture and measured size of every arm. † as in Table 1. † indicates a multi-domain aware variant. All four DiGress-family arms and DiG^2 also receive RRWP features (k=20); Pard, GraphRNN and G^{2}PT ignore them.

Model	Generative process	Model-specific hyperparameters	Domain conditioning
DeFoG†	discrete flow matching (CTMC)	marginal prior; stochasticity η=0, target guidance ω=0; `polydec` time distortion (train and sample); T=200; edge loss weight 5	one-hot, broadcast to node, edge and global channels
GraphBSI†	Bayesian sample inference	β: 3 $\rightarrow$ 20, β_0=1; γ=200 (tree 100); ρ=1; OU sampler (EM for planar, SBM, tree); T=200; edge loss weight 5	one-hot, broadcast to node, edge and global channels
DiGress†	discrete diffusion	marginal transitions; cosine schedule; 200 steps (train and sample); edge loss weight 5	one-hot prompt embedding, projected into node and edge channels
LGGM	discrete diffusion	as DiGress†	768-d BERT text-prompt embedding, same pathway
HOG-Diff†	two-phase higher-order score SDE	phase 1 VPSDE, linear $\beta \in [0.1, 1.0]$; phase 2 OU bridge, cosine, $\sigma_{\max}$=1.0; 200 scales per phase; random-walk depth 16; spectral basis from a 512-graph donor pool	additive bias on the time embedding and node pathway
GDSS†	joint score SDE on $(\boldsymbol{X}, \boldsymbol{A})$	$\boldsymbol{X}$: VP, $\beta \in [0.1, 1.0]$; $\boldsymbol{A}$: VE, $\sigma \in [0.2, 1.0]$; 200 scales; Euler predictor + Langevin corrector (SNR 0.1, scale 1.0, 1 step); attention channels $2/8/4$, 3 linears	additive bias after the first layer of both score nets
Pard†	block-wise autoregressive diffusion	degree-ordered blocks, max hops 1; 50 diffusion steps per block; two-stage training (block-size predictor separate); node count bounded by the domain's size prior	one-hot through Pard's conditioning blocks, node and edge
LGD†	latent diffusion	32-d latent per node and edge; x_0-parameterized DDPM, 1000 training steps; DDIM 200 at sampling	domain code from the frozen autoencoder's label embedder, cross-attention (dim 96)
GraphRNN†	autoregressive, BFS adjacency sequence	max previous nodes 186; generation cap 200 nodes; embedding 384	one-hot initializes the graph-level hidden state
G2PT	autoregressive, graph tokens	BFS edge-list tokenization; vocabulary 229; context 5120; node permutation augmentation; node count predicted; temperature 1.0, no top-k	native per-domain prefix token
SPECTRE†	spectrum-first WGAN-GP	eigenvalues $\rightarrow$ eigenvectors $\rightarrow$ adjacency; k=4 eigenpairs; noise dim 100; gradient penalty 5; 3 SON rotations	one-hot concatenated to every noise MLP input

Table 9: Parameters of each arm's generative process, excluding the backbone (Table 8) and the shared training protocol. Values follow each method's published configuration except where noted; DiG^2's are in Table 5. †Domain conditioning was added by us, because the upstream method is unconditional or conditions only on a global vector.

E ABLATION STUDY: EXTRA DETAILS

EMA prototypes ensure structural separability. Removing the EMA prototype embedding produces the largest overall structural degradation in the current ablation. PLANAR accuracy falls from 100.0% to 77.5%, while SBM drops from 85.0% to 42.5%. Their MMD Ratios simultaneously increase from 2.30 to 7.65 and from 1.80 to 4.82. Molecular validity also falls by 7.44 points on QM9 and 3.54 points on ZINC. The effect is therefore not limited to one graph family, a dedicated prototype representation is needed for the shared denoiser to keep heterogeneous modes separated.

Domain-specific priors prevent early cross-domain mixing. Replacing the per-domain corruption endpoints with a shared prior reduces PLANAR and SBM accuracy by 17.5 and 15.0 points and approximately doubles their MMD Ratios. The degradation persists on QM9 and ZINC, where validity falls by 6.06 and 3.04 points. Since this intervention changes the starting distribution rather than the denoiser architecture, the result indicates that domain separation is already consequential at the noisy endpoint, meaning the reverse process is easier to control when it begins from a prior matched to the requested domain.

Vocabulary restriction primarily protects distributional quality. Removing the per-domain cross-entropy mask causes smaller accuracy losses than removing the EMA embedding, but sharply worsens structural distribution matching. PLANAR retains 90.0% accuracy while its MMD Ratio rises from 2.30 to 6.37; SBM similarly moves from 1.80 to 3.48. This distinction is useful. Structural validity can remain high even when probability mass leaks onto classes outside the domain's support. Restricting the training and sampling support therefore acts as a semantic constraint rather than merely an optimization convenience.

Multi-depth denoising matters most for structurally difficult domains. Removing the depth mixture has little effect on the current molecular validity numbers, but degrades SBM and TREE accuracy by 12.5 and 7.5 points and raises their MMD Ratios from 1.80 to 5.13 and from 1.56 to 2.78. This pattern is consistent with the motivation for exposing reconstructions at multiple transformer depths, different graph families require distinctive denoising effort. Taken together, the ablation results indicate that DiG^2 does not obtain its performance from a single conditioning trick. Domain-specific endpoints, support constraints, internal conditioning, and depth adaptation make complementary contributions to maintaining separation.

F α-INTERPOLATION: EXTRA DETAILS

Baseline Testing Details. DiG^2 and LGGM generate at node counts drawn from the target band's histogram. GenMol controls size through its token-length pool, DemoDiff cannot condition on size, so its molecules are filtered to the target band after sampling. We therefore distinguish *raw validity*, the fraction of all attempts that sanitize, from *in-band yield*, the fraction of attempts that are both valid and inside the target band; for size-conditioned generators the two coincide.

F.1 ZERO-SHOT VALIDITY OF α-INTERPOLATED MOLECULES

Zero-shot validity. DiG^2 generates valid molecules along the entire path. On GuacaMol 10–14 its validity is 0.82 at the QM9 endpoint and 0.95–0.96 for every $\alpha \geq 0.5$. LGGM, whose prompt blend is the closest mechanism to ours, collapses away from ZINC: 0.17 at $\alpha = 0$ and 0.32 at $\alpha = 0.5$ on GuacaMol 10–14, and 0.06–0.07 on MOSES 15–19 for $\alpha \leq 0.5$. DemoDiff's raw validity (0.85–0.93) is close to DiG^2's, but because it cannot condition on size, only 15–28% of its attempts land in GuacaMol 10–14 and 28–39% in MOSES 15–19. The one region where DiG^2 is weaker is the QM9 endpoint on MOSES 15–19 (0.56): QM9 conditioning asked to build molecules twice QM9's size limit.

Secondary metrics. Tables below list every cell. DemoDiff's uniqueness is computed over all generated SMILES rather than its deduplicated output.

Property-dial cells. Tables 12 and 13 list every cell behind Figures 3 and 4. The plotted values are taken from each figure's build record, so they match the curves exactly.

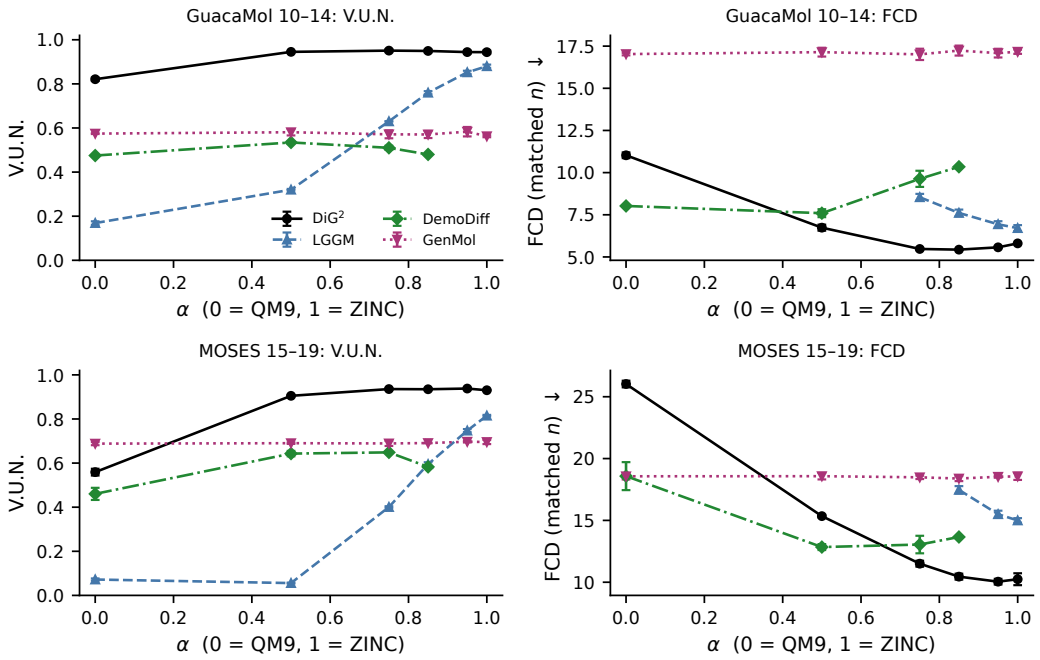


Figure 6: **Zero-shot molecule generation along the QM9→ZINC axis (size-conditioned protocol).** *Left:* raw validity. DemoDiff and GenMol reach the band by rejection, so their raw validity is measured on the unfiltered pool and is shared across bands; their in-band yield is given in the text and Appendix F. *Right:* FCD to the target band. Baselines are subsampled to DiG^2's valid-molecule count in each cell. Cells with fewer than 1,000 valid molecules are omitted (LGGM at low α). Mean $\pm$ s.d. over seeds. No target molecule is used for training or tuning.

Table 10: Zero-shot cells, GuacaMol 10–14 (mean over seeds). QED/SA W_1 are Wasserstein distances to the target's property distribution (lower is closer).

Model	α	seeds	n_{valid}	Valid	Uniq	Novel	Filters	QED W_1	SA W_1	SNN
DiG^2	0.00	3	1643	0.822	1.000	1.000	0.584	0.083	2.219	0.235
DiG^2	0.50	3	1896	0.948	1.000	0.997	0.654	0.032	1.266	0.268
DiG^2	0.75	3	1916	0.958	1.000	0.993	0.705	0.040	0.798	0.289
DiG^2	0.85	3	1915	0.958	1.000	0.992	0.720	0.050	0.680	0.295
DiG^2	0.95	3	1908	0.954	1.000	0.990	0.735	0.056	0.574	0.300
DiG^2	1.00	3	1905	0.953	0.999	0.991	0.738	0.061	0.531	0.301
LGGM	0.00	3	337	0.169	1.000	1.000	0.770	0.073	2.209	0.252
LGGM	0.50	3	640	0.320	1.000	0.999	0.758	0.056	2.341	0.236
LGGM	0.75	3	1262	0.631	1.000	0.998	0.725	0.040	1.528	0.256
LGGM	0.85	3	1525	0.763	1.000	0.997	0.725	0.037	1.282	0.265
LGGM	0.95	3	1711	0.856	1.000	0.996	0.725	0.037	1.011	0.275
LGGM	1.00	3	1766	0.883	1.000	0.996	0.730	0.040	0.905	0.279
DemoDiff	0.00	1	2007	0.850	0.602	0.928	0.791	0.035	0.297	0.387
DemoDiff	0.50	2	2005	0.892	0.659	0.909	0.773	0.042	0.394	0.399
DemoDiff	0.75	2	2005	0.915	0.615	0.907	0.817	0.066	0.421	0.399
DemoDiff	0.85	1	2002	0.930	0.576	0.895	0.791	0.060	0.404	0.398
GenMol	0.00	3	1035	0.999	0.639	0.900	0.777	0.088	0.851	0.449
GenMol	0.50	3	1025	0.999	0.646	0.900	0.771	0.088	0.834	0.446
GenMol	0.75	3	1034	0.999	0.643	0.889	0.773	0.087	0.829	0.444
GenMol	0.85	3	1022	0.999	0.640	0.893	0.765	0.086	0.840	0.448
GenMol	0.95	3	1033	0.999	0.647	0.902	0.786	0.088	0.839	0.446
GenMol	1.00	3	1018	0.999	0.633	0.889	0.785	0.089	0.842	0.449

F.2 DiG^2 QED Generated Samples

Table 11: Zero-shot cells, MOSES 15–19 (mean over seeds). QED/SA W_1 are Wasserstein distances to the target's property distribution (lower is closer).

Model	α	seeds	n_{valid}	Valid	Uniq	Novel	Filters	QED W_1	SA W_1	SNN
DiG^2	0.00	3	1118	0.559	1.000	1.000	0.355	0.249	3.287	0.230
DiG^2	0.50	3	1811	0.905	1.000	0.999	0.489	0.125	1.957	0.291
DiG^2	0.75	3	1874	0.937	1.000	0.999	0.601	0.075	1.370	0.328
DiG^2	0.85	3	1877	0.939	1.000	0.996	0.635	0.065	1.169	0.340
DiG^2	0.95	3	1884	0.942	1.000	0.995	0.666	0.058	1.004	0.350
DiG^2	1.00	3	1872	0.936	1.000	0.994	0.672	0.056	0.973	0.353
LGGM	0.00	3	144	0.072	1.000	1.000	0.556	0.155	3.406	0.223
LGGM	0.50	3	111	0.056	1.000	1.000	0.538	0.143	3.729	0.230
LGGM	0.75	3	801	0.401	1.000	1.000	0.460	0.097	2.513	0.268
LGGM	0.85	3	1190	0.595	1.000	1.000	0.479	0.088	2.180	0.279
LGGM	0.95	3	1495	0.748	1.000	0.999	0.526	0.085	1.809	0.293
LGGM	1.00	3	1630	0.815	1.000	1.000	0.551	0.084	1.659	0.298
DemoDiff	0.00	3	2000	0.858	0.540	0.993	0.636	0.116	0.324	0.372
DemoDiff	0.50	3	2000	0.896	0.728	0.987	0.687	0.092	0.231	0.426
DemoDiff	0.75	2	2000	0.915	0.718	0.987	0.690	0.086	0.208	0.424
DemoDiff	0.85	1	2000	0.930	0.634	0.987	0.661	0.094	0.248	0.427
GenMol	0.00	3	1261	0.996	0.744	0.929	0.700	0.031	0.593	0.503
GenMol	0.50	3	1252	0.996	0.746	0.929	0.691	0.031	0.583	0.505
GenMol	0.75	3	1264	0.996	0.743	0.931	0.694	0.031	0.577	0.504
GenMol	0.85	3	1272	0.995	0.746	0.930	0.698	0.031	0.577	0.504
GenMol	0.95	3	1257	0.997	0.742	0.943	0.695	0.030	0.583	0.503
GenMol	1.00	3	1245	0.995	0.748	0.934	0.692	0.030	0.583	0.505

Table 12: Per-cell values behind Figure 3 (GuacaMol band 15–19). QED is the mean over a cell's molecules, then mean $\pm$ s.d. over seeds, as plotted (a single-seed cell shows its value alone). n / seed: molecules scored per seed. DiG^2 writes 500 generations per seed; DemoDiff's cells are resampled to the reference heavy-atom composition and GenMol's are filtered to 15–19 heavy atoms, so their n is what remains. The α^* rows are the calibrated DiG^2 cells for the p25 (0.6225) and p50 (0.738) targets. p75 (0.8222) is out of reach. The figure draws $0 \le \alpha \le 1$ (DiG^2's $\alpha = 0$ lies below its axis); the $\alpha > 1$ cells are listed for completeness.

Model	α	seeds	n / seed	heavy atoms	QED
DiG^2	0.00	5	500	17.4	0.559 ± 0.006
DiG^2	0.25	5	500	17.5	0.620 ± 0.006
DiG^2	0.50	5	500	17.5	0.684 ± 0.003
DiG^2	0.75	5	500	17.5	0.732 ± 0.003
DiG^2	0.85	5	500	17.5	0.746 ± 0.004
DiG^2	0.95	5	500	17.5	0.753 ± 0.006
DiG^2	1.00	5	500	17.5	0.758 ± 0.007
DiG^2	1.25	5	500	17.5	0.763 ± 0.006
DiG^2	1.50	5	500	17.5	0.762 ± 0.003
DiG^2	2.00	5	500	17.5	0.715 ± 0.002
DiG^2, α^* (target 0.6225)	0.267	3	500	17.5	0.627 ± 0.005
DiG^2, α^* (target 0.738)	0.793	3	500	17.5	0.739 ± 0.003
DemoDiff	0.00	3	1051	17.5	0.704 ± 0.015
DemoDiff	0.50	3	1527	17.5	0.735 ± 0.014
DemoDiff	0.75	2	1471	17.5	0.741 ± 0.004
DemoDiff	0.85	1	1457	17.5	0.733
GenMol	0.00	3	1476	17.2	0.803 ± 0.002
GenMol	0.50	3	1502	17.2	0.802 ± 0.001
GenMol	0.75	3	1500	17.2	0.803 ± 0.002
GenMol	0.85	3	1458	17.2	0.803 ± 0.001
GenMol	0.95	3	1475	17.2	0.804 ± 0.001
GenMol	1.00	3	1454	17.2	0.803 ± 0.001

F.3 MULTI-DOMAIN SAMPLES

Table 13: Per-cell values behind Figure 4 (MOSES band 20–27). QED and logP are means over a cell's molecules, then mean ± s.d. over seeds (a single-seed cell shows its value alone). DiG^2: n is the valid molecules per seed out of 2,000 generated, and Valid is their fraction. †below the 0.70 validity floor, generated but not drawn. DemoDiff: n after resampling to the reference heavy-atom composition. GenMol: n after filtering to 20–27 heavy atoms.

Model	α	seeds	n / seed	Valid	heavy atoms	QED	logP
DiG^2	0.00	3	1604	0.802	22.6	0.580 ± 0.003	0.701 ± 0.028
DiG^2	0.50	3	1819	0.909	22.6	0.738 ± 0.004	1.958 ± 0.015
DiG^2	0.75	3	1851	0.925	22.6	0.776 ± 0.005	2.432 ± 0.022
DiG^2	0.85	3	1863	0.931	22.6	0.780 ± 0.001	2.581 ± 0.022
DiG^2	0.95	3	1853	0.926	22.6	0.789 ± 0.002	2.756 ± 0.023
DiG^2	1.00	3	1860	0.930	22.6	0.791 ± 0.001	2.830 ± 0.026
DiG^2	1.25	3	1817	0.908	22.6	0.790 ± 0.003	3.217 ± 0.018
DiG^2	1.50	3	1779	0.889	22.6	0.775 ± 0.002	3.632 ± 0.046
DiG^2	1.75	3	1625	0.812	22.6	0.733 ± 0.002	4.163 ± 0.050
DiG^2	2.00†	3	1335	0.668	22.6	0.652 ± 0.003	4.734 ± 0.049
DemoDiff	0.50	3	851		22.6	0.673 ± 0.018	3.670 ± 0.200
DemoDiff	0.75	2	809		22.6	0.677 ± 0.018	3.549 ± 0.015
DemoDiff	0.85	1	946		22.6	0.686	3.357
GenMol	0.00	3	1192		23.3	0.689 ± 0.007	2.630 ± 0.031
GenMol	0.50	3	1178		23.3	0.693 ± 0.012	2.672 ± 0.077
GenMol	0.75	3	1136		23.3	0.691 ± 0.006	2.678 ± 0.014
GenMol	0.85	3	1173		23.3	0.693 ± 0.001	2.690 ± 0.020
GenMol	0.95	3	1160		23.2	0.694 ± 0.003	2.651 ± 0.055
GenMol	1.00	3	1161		23.2	0.697 ± 0.000	2.606 ± 0.025

G Initial Investigation: Graph-Level Conditioning

This section details results from our initial investigation into promoting multi-domain awareness in single-domain graph generation models. Within Table 14, we demonstrate these architectures carry conditioning in a single graph-level vector $\boldsymbol{y}$. Perturbing the domain one-hot in $\boldsymbol{y}$ moves the network logit output ($\hat{\boldsymbol{X}}$) by at most, $\max|\Delta\hat{\boldsymbol{X}}| = 0.06$. Whereas, perturbing the node features $\boldsymbol{X}$ moves it by 5.14. The conditioning pathway is roughly $85\times$ weaker than the data pathway, making it too weak to override an 839:1 prior. LGGM, on identical data, does not suffer this, given its native pathway broadcasts the conditioning vector onto every node and edge before the first layer.

Domain	n (mean)	Avg. degree	Node-class histogram
Graph-level conditioning only			
planar	64.0	15.61	C 0.87, N 0.08, S 0.04, O 0.02
sbm	115.4	50.38	C 0.63, `generic` 0.24, N 0.06, O 0.03
tree	64.0	15.41	C 0.85, N 0.10, O 0.03, S 0.03
Broadcast to node and edge features			
planar	64.0	4.77	`generic` **1.00**
sbm	97.6	9.75	`generic` 0.29, C 0.21, N 0.16, O 0.16
tree	64.0	2.15	`generic` **1.00**

Table 14: *Why do baseline models fail at multi-domain graph generation?* The following table shows the emitted node classes from a baseline model, before and after the conditioning signal is broadcast to node and edge features. The structural domains use only the `generic` class, so any atom mass is conditioning leakage from the pooled molecular prior. Node counts come from a histogram prior in both settings and are correct throughout, which localizes the failure to graph content. Here we see the graph-level conditioning signal unable to prevent molecular vocabulary (i.e. C,N,S,O) from leaking into the `generic` class which PLANAR, SBM, and TREE require.

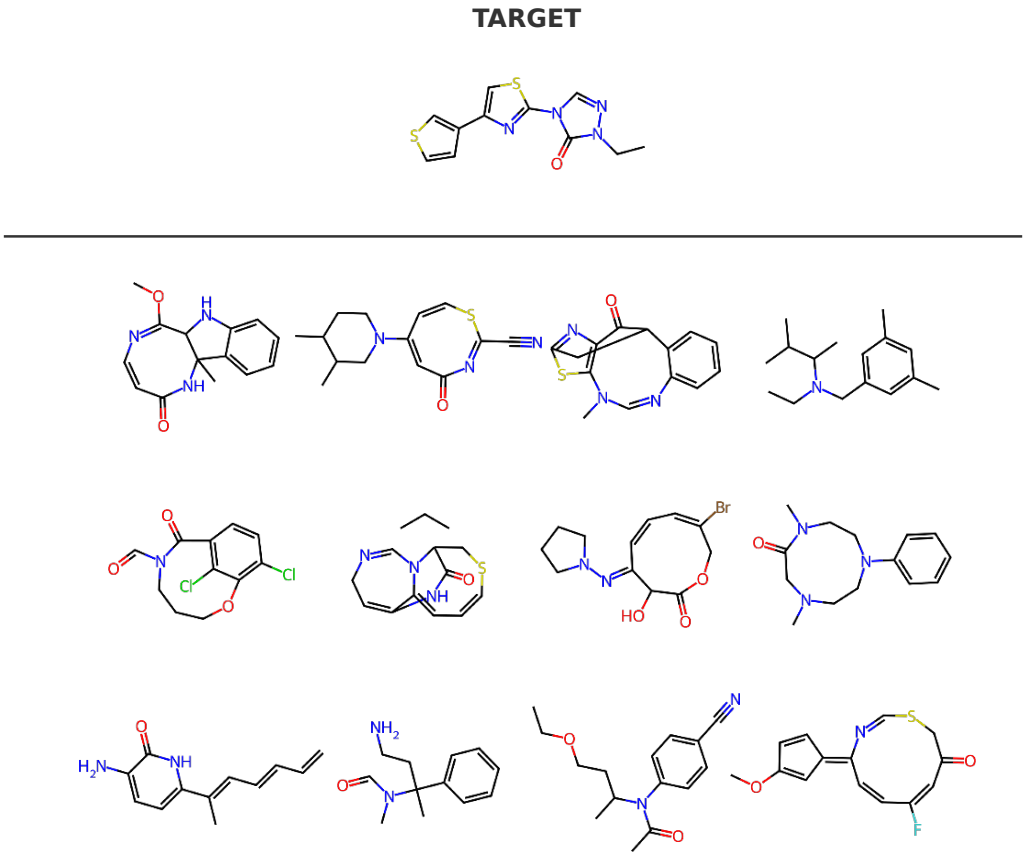


Figure 7: **QED target - DiG^2 Samples.** Top: the real GuacaMol band-15–19 molecule at the p50 target. Below the rule: DiG^2 generations at $\alpha^* = 0.793$, the twelve closest to the target.

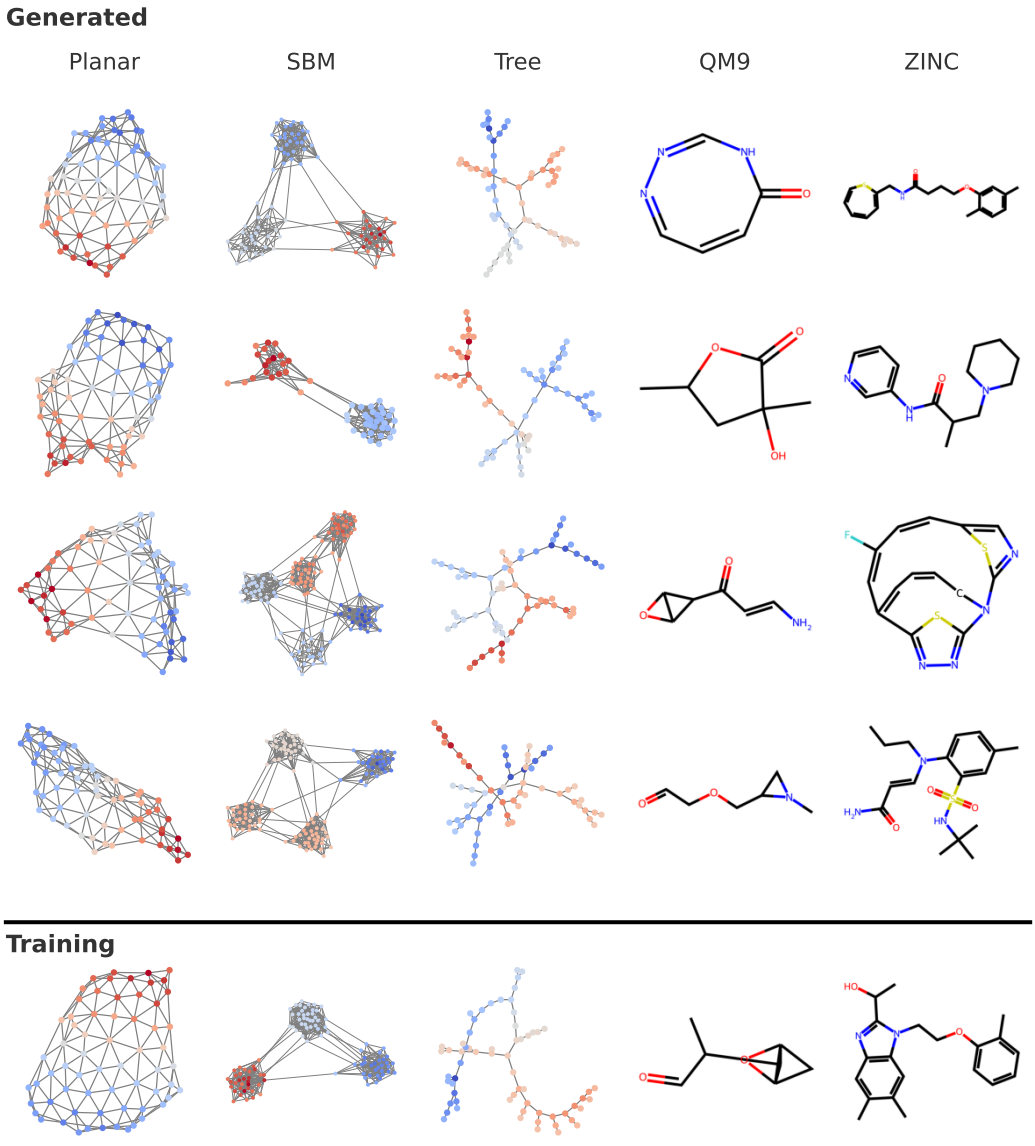


Figure 8: **DiG^2 generations across all five domains, over real training graphs.** One column per domain. The top four rows are generations from a single DiG^2 checkpoint, the bottom row (below the rule) is a reference graph from the corresponding domain.