Flowette: Flow Matching with Graphette Priors for Graph Generation

Asiri Wijesinghe[1*] **Sevvandi Kandanaarachchi**[1] **Daniel M. Steinberg**[1]
Cheng Soon Ong[1,2]
[1]CSIRO [2]Australian National University

Abstract

We study generative modeling of graphs with recurring subgraph motifs. We propose Flowette, a continuous flow matching framework that employs a graph neural network-based transformer to learn a velocity field over graph representations with node and edge attributes. Our model promotes topology-aware alignment through optimal transport-based coupling and preserves structural fidelity through endpoint- and chemistry-aware regularisation. To incorporate domain-driven structural priors, we introduce graphettes, a new probabilistic family of graph structure models that generalize graphons via controlled structural edits for motifs such as rings, stars, and trees. We theoretically analyze the coupling, invariance, and structural properties of the framework, evaluate it on synthetic and molecular benchmarks, and isolate the contributions of the structural prior, the optimal-transport coupling, and the regularisation terms through controlled ablations. Flowette achieves competitive performance overall, attaining state-of-the-art results on several metrics across multiple benchmarks, highlighting the effectiveness of combining structural priors with flow-based training for modeling complex graph distributions.

1 Introduction

Generative modeling of graphs has applications spanning molecular design, social networks, biological systems, and combinatorial optimization [73, 35, 75]. Graphs exhibit complex structure, permutation symmetry, and rich higher-order motifs such as cycles, hubs, and repeated subgraphs. These motifs encode essential domain semantics. For instance, molecular graphs are characterized by ring structures and valence constraints, social networks contain hub-and-spoke patterns, and many biological and combinatorial graphs are tree-like. Capturing such recurring substructures within a single framework, rather than via motif-specific architectures, remains a major open challenge for graph generative models. Existing approaches rely on domain-specific architectures, resulting in specialized models for each motif [30, 50, 33]. This limits transferability and obscures the underlying principles of structure-aware graph generation.

Recent progress in continuous-time generative modeling-notably diffusion models and flow matching-has provided powerful tools for high-dimensional generation [61, 41]. Flow matching (FM) directly learns a velocity field transporting samples from a source distribution to the data distribution [41, 46]. However, existing graph FM approaches typically rely on implicit or independent coupling between noise and data graphs, ignoring structural compatibility under permutation and size variability. As a result of forcing models to interpolate between topologically mismatched graphs, velocity supervision can be high-variance and inconsistent.

Three ingredients are essential for structure-aware flow-based graph generation. First, supervision pairs in flow matching should be structurally aligned. Second, the training objective must remain

*corresponding author: asiriwijesinghe.wijesinghe@csiro.au

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

reliable under finite-step integration, where local velocity errors can otherwise compound into invalid endpoints, while also respecting domain feasibility. Third, the source distribution itself should encode meaningful structural priors, reflecting the dominant motifs.

We introduce Flowette, a graph generation framework that contains these three ingredients. **(1)** Flowette couples noise graphs and data graphs using Fused Gromov-Wasserstein (FGW) optimal transport [64], yielding node permutation-consistent correspondences across variable-sized graphs. By providing supervision pairs that respect topology, we enable structure aligned flow matching on graphs. The velocity field is parameterized by a topology respecting Graph Neural Network (GNN) based transformer that jointly evolves node features, edge features, and adjacency values. **(2)** We introduce regularisation terms in the training objective to preserve long range structures and permutation equivariance. The resulting training objective leads to more stable continuous time generation and improved structural fidelity. **(3)** To enable structural inductive biases in the source distribution, we introduce a new mathematical object *graphettes*, a probabilistic family of graph priors that generalize graphons via controlled edits. Graphettes allow principled injection or removal of motifs (rings, stars, cycles), providing a unified mechanism to model dense, sparse, and motif-rich graphs within a single framework. Unlike task-specific heuristics, graphettes decouple structural assumptions from the neural architecture and require only coarse structural knowledge of the target family (whether it contains rings, hubs, or trees), rather than empirical tuning of model components. We empirically validate that each of these three ingredients contributes independently to performance: replacing the graphette prior with a standard graphon, replacing FGW coupling with random or Euclidean pairing, or removing individual regularisation terms each produces substantial degradation across both synthetic and molecular benchmarks.

2 Background and setting

Graphs have interchangeable representations, and we use the following conventions. A graph G with n vertices (nodes) can be written in terms of vertices and edges, (V, E), or equivalently by its $A \in \{0,1\}^{n\times n}$ binary adjacency matrix. We retrieve the sets of vertices and edges in G by $V(G)$ and $E(G)$ respectively and denote the number of vertices by $|V(G)|$.

When we need to model attributed graphs with node and edge features, we use $X \in \mathbb{R}^{n\times d_x}$ for node features and $F \in \mathbb{R}^{n\times n\times d_f}$ for edge features of G. Sec. 3 introduces the flow matching framework for attributed graphs, and we denote a graph G by (A, X, F), a triple consisting of the adjacency matrix, node and edge features of G. In Sec. 4 we introduce graphette priors focusing on non-attributed graphs G, which we denote by (V, E). We use $G \sim \mathcal{W}$ to indicate that a graph is sampled from some distribution or using a sampling function $\mathcal{W}$.

2.1 Flow Matching on Graphs

Related Work. Graph generation needs to capture local and global structures, as motivated by domains like molecular design, combinatorial optimisation, and biological networks [52, 63, 72]. Existing methods can be broadly grouped into autoregressive [73, 39, 50, 23] and one-shot paradigms [15, 58, 33, 16]. Diffusion-based generative models have recently become the dominant one-shot paradigm for graphs [53, 32], but discrete time versions tie sampling trajectory to the training schedule. Continuous-time discrete diffusion models based on Continuous Time Markov Chains (CTMCs) address this rigidity by decoupling the sampling grid from the training objective [9, 71]. DisCo [71], for instance, defines a discrete-state continuous-time diffusion on graphs and inherits the permutation-equivariant/invariant properties of DiGress [67] while enabling flexible numerical integration and theoretical control of approximation error [71]. Discrete flow matching (DFM) extends FM to discrete state spaces, where probability paths are defined by time-inhomogeneous CTMCs and the model parameterises transition rates rather than continuous velocities [10, 22, 55]. In parallel, some works have explored continuous FM and flow-based formulations for graphs by relaxing discrete structures into continuous spaces [44, 19]. However, they typically rely on implicit couplings between source and target graphs. Concurrent work [12] independently explores domain-informed source distributions for graph flow matching, focusing on graph reconstruction rather than generation. Extended background on flow matching, together with a more thorough discussion of related work, is provided in Appendix B.

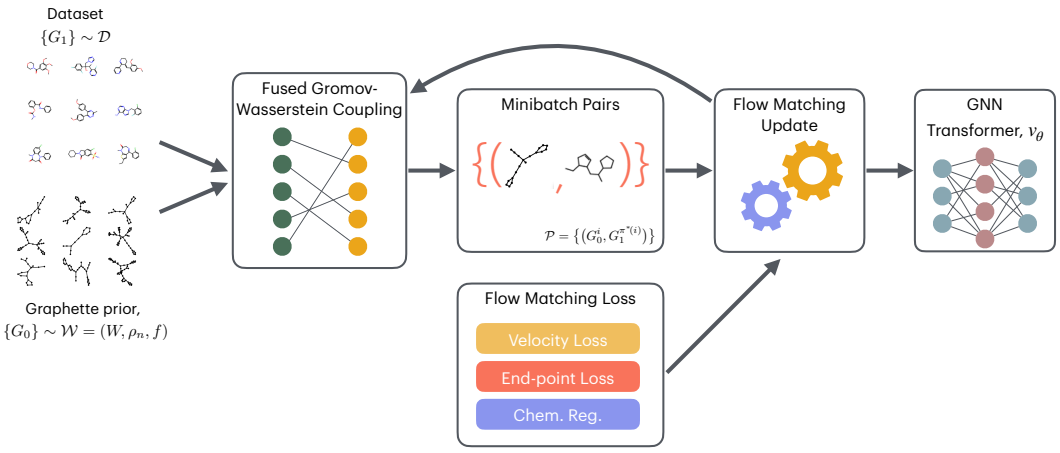

Figure 1: Flowette training scheme. FGW optimal transport with Hungarian matching discussed in Sec. 3.1, GNN Transformer in App. C, Topology-aware loss and regularisation in Sec. 3.3 and graphettes in Sec. 4. See Algorithm 1 for details.

Flow matching offers an alternative to diffusion-based training for continuous-time generative models [41]. Rectified-flow variants adopt linear interpolants, and regress the associated constant conditional velocity field [46], leading to state-of-the-art results in high-dimensional image and sequence generation [20, 49]. We need coupling between graph samples G_0^i from a source (noise) distribution in $\mathcal{S}$ and target (data) graphs G_1^j from the data distribution in $\mathcal{S}$. $\mathcal{S}$ will later correspond to a continuous relaxation of adjacency matrices, node features, and edge features. FM learns a time-conditioned vector field $v_\theta : \mathcal{S} \times [0,1] \to \mathcal{S}$, parameterized by θ, that approximates the conditional velocity along the path defined by time $t \in [0,1]$. It assumes paired samples (G_0^i, G_1^j) together with interpolating states $(G_t)_{t\in[0,1]}$ defining a probability path between the noise distribution and the data distribution. The FM training objective is defined as a regression loss:

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{\substack{G_0,G_1 \\ t\sim\mathcal{U}[0,1]}} \left[\| v_\theta(G_t, t) - u^\star(G_t \mid G_0, G_1) \|_2^2 \right]. \tag{1}$$

At generation time, new samples are obtained by solving the ordinary differential equation $\frac{dG_t}{dt} = v_\theta(G_t, t)$, where $G_{t=0} \sim p_0$, using a numerical integrator (e.g. Euler), and taking $G_{t=1}$ as a sample from the learned model distribution.

2.2 Structural priors

To provide suitable inductive bias for a particular application, we need a flexible class of probabilistic models to design meaningful priors. This work extends graphons, a widely used approach for graph priors [27].

Definition 2.1. (**Graphon**) A *graphon* is a symmetric, measurable function $W : [0,1]^2 \to [0,1]$. Given $n \in \mathbb{N}$, a graph G with n nodes can be sampled from W by: (i) drawing latent variables $x_1, \ldots, x_n \overset{iid}{\sim} \mathcal{U}[0,1]$, and (ii) for each $1 \le i < j \le n$, sampling an edge $A_{ij} \sim \mathrm{Bernoulli}(W(x_i, x_j))$ independently, then set $A_{ji} = A_{ij}$. We write $G \sim \mathsf{G}(n, W)$.

We give an accessible introduction to graphons, sparsified graphons, and graphexes (the structural priors that graphettes generalise) in Appendix A. The limitation of graphons is that graphs sampled from graphons are dense, i.e., the number of edges grow quadratically with the number of nodes. However, many real-world graphs such as social and molecular networks are sparse, i.e., the number of edges grow subquadratically with the nodes. Sparsified graphons [34, 36] is an alternative used to model sparse graphs.

Definition 2.2. (**Sparsified graphon**.) Let $W : [0,1]^2 \to [0,1]$ denote a graphon and $\{\rho_i\}_i = (\rho_1, \rho_2, \ldots, \rho_n, \ldots)$ denote a real-valued sequence converging to zero such that $0 \le \rho_i \le 1$. Then the graphon $W_n := \rho_n W$ is called the sparsified graphon. We denote a graph G with n nodes sampled from W_n by $G \sim \mathsf{G}(n, W, \rho_n)$.

3 Structure Preserving Flows

We visualise our overall design in Figure 1. In this section we address key challenges that arise when learning continuous-time generative dynamics on graphs and clarify how our method departs from the standard flow-matching pipeline. First, naively coupling noise and data graphs (e.g., by minibatch index) ignores structural compatibility and leads to high-variance supervision since the velocity field must interpolate between topologically mismatched source-target pairs. Second, unconstrained velocity fields defined over continuous graph representations can yield invalid structures under finite-step integration, particularly for molecular graphs where chemical constraints must be respected. To address these issues, we have three core components: (i) a structure-aware coupling mechanism for supervision (Sec. 3.1), which replaces independent (random) coupling or Euclidean optimal transport coupling with FGW-distance and Hungarian graph matching; (ii) a permutation-equivariant GNN transformer that parameterizes the velocity field over evolving graph representations (Sec. 3.2 and App. C); and (iii) an enhanced training objective (Sec. 3.3) that augments the flow-matching loss with endpoint consistency and chemistry-aware structural regularization. Together, these design choices yield coherent, permutation-invariant supervision across variable-sized graphs and promote structurally valid generations under finite-step integration. In this section we use G to refer to attributed graphs, $G = (A, X, F)$ and we present the full training scheme in Algorithm 1.

3.1 Graph Structure Coupling, (G_0, G_1)

For our graphs we extract structural node embeddings $Z = [z_1, \ldots, z_n]^\top = f_{\text{enc}}(G) \in \mathbb{R}^{n \times d_z}$, using a pretrained edge-aware Graph Isomorphism Network (GIN) encoder f_{enc} [70]. The encoder is a lightweight graph autoencoder trained on the training split only; never on validation or test data, and frozen thereafter. It contributes structural embeddings to the FGW cost but does not influence the velocity field, integration, or generated outputs (full details and a sensitivity ablation in Appendix J.9.2).

FGW-Distance Computation. Let $G_0 = (A_0, X_0, F_0)$ and $G_1 = (A_1, X_1, F_1)$ denote two undirected attributed graphs with $|V(G_0)| = n_0$ and $|V(G_1)| = n_1$, respectively. Let $Z_0 = f_{\text{enc}}(G_0)$ and $Z_1 = f_{\text{enc}}(G_1)$ denote the corresponding encoder-induced structural node embeddings. We define the intra-graph structural cost matrices $C(Z_0) \in \mathbb{R}^{n_0 \times n_0}$ and $C(Z_1) \in \mathbb{R}^{n_1 \times n_1}$ by $C(Z_a)_{ik} = \|z_i^a - z_k^a\|_2^2$, where $a \in \{0, 1\}$. The cross-graph feature cost matrix $M(X_0, X_1) \in \mathbb{R}^{n_0 \times n_1}$ is defined as $M(X_0, X_1)_{ij} = \|x_i^0 - x_j^1\|_2^2$. Using these costs, we can compute a fused optimal transport plan $T^\star \in \mathbb{R}^{n_0 \times n_1}$ by solving,

$$T^\star \in \underset{T \in \Pi(p,q)}{\arg\min} \big(1 - \alpha\big) \langle T, M(X_0, X_1) \rangle + \alpha \, \mathcal{L}_{\text{GW}}\big(T;\, C(Z_0), C(Z_1)\big), \tag{2}$$

where $\alpha \in [0, 1]$ trades off features and structure, $\Pi(p, q) = \{T \in \mathbb{R}_{\geq 0}^{n_0 \times n_1} : T\mathbf{1} = p,\ T^\top \mathbf{1} = q\}$. The corresponding FGW distance [64] is the optimal value of the fused cost minimized in Eq. (2), i.e.,

$$\text{FGW}_\alpha(G_0, G_1) := \min_{T \in \Pi(p,q)} \big(1 - \alpha\big) \langle T, M(X_0, X_1) \rangle + \alpha \, \mathcal{L}_{\text{GW}}\big(T;\, C(Z_0), C(Z_1)\big), \tag{3}$$

where the squared-loss GW term is $\mathcal{L}_{\text{GW}}\big(T; C(Z_0), C(Z_1)\big) = \sum_{i,k=1}^{n_0} \sum_{j,\ell=1}^{n_1} \big(C(Z_0)_{ik} - C(Z_1)_{j\ell}\big)^2 T_{ij} \, T_{k\ell}$. We use uniform node measures $p \in \Delta_{n_0}$ and $q \in \Delta_{n_1}$, where $\Delta_n = \big\{r \in \mathbb{R}_{\geq 0}^n \mid \mathbf{1}^\top r = 1\big\}$ denotes the probability simplex on n elements, and $p = n_0^{-1} \mathbf{1}_{n_0}$ for G_0 and $q = n_1^{-1} \mathbf{1}_{n_1}$ for G_1. The resulting optimal plan $T^\star$ provides the node-level correspondence used to align the target graph to the source-node ordering, while $\text{FGW}_\alpha(G_0, G_1)$ denotes the corresponding optimal objective value and is used as the graph-to-graph discrepancy.

Batch-Level Coupling via Hungarian Matching. In each minibatch, we sample a set of B noise graphs $\{G_0^i\}_{i=1}^B \sim \mathcal{W}$ and B target graphs $\{G_1^j\}_{j=1}^B \sim \mathcal{D}$. We compute a cost matrix $D \in \mathbb{R}^{B \times B}$ with entries

$$D_{ij} = \text{FGW}(G_0^i, G_1^j), \tag{4}$$

then obtain a one-to-one bipartite assignment: $\pi^\star = \arg\min_{\pi \in \mathfrak{S}_B} \sum_{i=1}^B D_{i,\pi(i)}$, where $\mathfrak{S}_B$ is the set of permutations [38]. This prevents degenerate training in which many target graphs align to the same noise graph, and yields stable coverage across diverse graph topologies.

We support three coupling protocols differing in how often this batch-level FGW computation is performed. The default, used throughout Algorithm 1 and all main results, is *online coupling*: FGW costs and the Hungarian assignment are recomputed every minibatch. Two amortised variants – *fixed-pool* and *periodic* coupling – cache the correspondence across epochs and substantially reduce training cost (protocol details and a scalability study in App. J.8).

We refer to this Hungarian matching procedure on FGW-distances as ***FGW-coupling***. It ensures that (A_0, X_0, F_0) and (A_1, X_1, F_1) correspond in a structure-preserving manner, resulting in lower-variance velocity supervision than independent coupling or Euclidean optimal transport coupling.

Finally, in App. E.1 we show that; FGW-distance is permutation invariant (Lemma E.1), FGW-distance is 0 for isomorphic graphs (Corollary E.2), and FGW-coupling recovers the optimal permutation-consistent bipartite (Theorem E.3).

3.2 Flow Matching on FGW-Coupled Graphs.

For each FGW-coupled graph pair (G_0, G_1), we consider graph representations (A_0, X_0, F_0) and (A_1, X_1, F_1) restricted to a common node budget n under the structure-preserving node correspondence induced by FGW (the soft node correspondence and target-graph transport are given explicitly in App. I.1). In particular, the node budget is handled directly in our flow matching scheme by sampling noise graphs from the graphette prior with node counts explicitly matched to their paired target graphs, ensuring $|V(G_0)| = |V(G_1)|$ for every coupled pair. This avoids any size mismatch and removes the need for padding or truncation during interpolation. In this setting, $G_0 \sim \mathcal{W}$ denotes a noise graph sampled from the graphette prior (which we define in the next section), while $G_1 \sim \mathcal{D}$ denotes a target graph sampled from the data distribution. The prior $\mathcal{W}$ is used exclusively to generate the initial graph topology A_0. The X_0 and F_0 are then sampled independently from fixed categorical prior distributions corresponding to atom-type and bond-type priors, respectively. For $t \sim \mathcal{U}[0, 1]$, we construct rectified linear interpolants,

$$A_t = (1-t)A_0 + tA_1, \quad X_t = (1-t)X_0 + tX_1, \quad F_t = (1-t)F_0 + tF_1. \tag{5}$$

Under this parameterization, the ideal transport velocities are constant and given by $\Delta A = A_1 - A_0$, $\Delta X = X_1 - X_0$, and $\Delta F = F_1 - F_0$. We model the continuous-time transport dynamics using a time-conditioned velocity field, $v_\theta : (A_t, X_t, F_t, t) \mapsto \big(v_A,\ v_X,\ v_F\big)$, where $v_A \in \mathbb{R}^{n\times n}$, $v_X \in \mathbb{R}^{n\times d_x}$, and $v_F \in \mathbb{R}^{n\times n\times d_f}$ predict the instantaneous velocities of the adjacency, node features, and edge features, respectively. We use a novel *permutation equivariant* GNN transformer to implement the velocity field predictor, v_θ. We give details of its architecture in App. C as well as proof of its permutation equivariance. Unlike existing graph transformers that operate on discrete graph states and predict categorical outputs – DiGress [67] and DeFoG [55] parameterise categorical denoising distributions or CTMC transition rates, while VFM/CatFlow [19] induces its vector field from a predicted categorical endpoint – our transformer directly parameterises and integrates the coupled continuous velocities (v_A, v_X, v_F) over the evolving triple (A_t, X_t, F_t). App. C.2 details architectural distinctions underlying this difference. This velocity field predictor is trained via a continuous flow-matching objective, matching (v_A, v_X, v_F) to the rectified displacements $(\Delta A, \Delta X, \Delta F)$ at intermediate times by minimizing a flow matching loss (Eq. 6). Although the underlying graphs are discrete, all training and integration are performed in continuous space; discreteness is enforced only at generation time via projection (see Appendix J.1).

3.3 Flow Matching Learning Objective

We propose a rectified flow-matching objective for graph generation that learns a continuous-time transport from graphette priors to the data distribution while preserving both local and global graph structure. Since a fixed velocity residual induces a larger predicted endpoint error when incurred early in the trajectory than late (App. E.3.1), locally accurate velocity predictions on average do not guarantee low endpoint error under finite-step solvers, an issue exacerbated in graph domains. We therefore couple local velocity matching with an endpoint-aware reweighting of that same residual, together with domain-specific regularization, to ensure stable and structurally valid graph generation. Let $\beta_{\text{end}}, \beta_{\text{val}}, \beta_{\text{atom}} \geq 0$ be scalar weighting coefficients. The overall training objective is as follows where each term is defined below.

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{vel}} + \beta_{\text{end}} \cdot \mathcal{L}_{\text{end}} + \beta_{\text{val}} \cdot \mathcal{L}_{\text{val}} + \beta_{\text{atom}} \cdot \mathcal{L}_{\text{atom}}, \tag{6}$$

Algorithm 1 Flowette Flow Matching Training

input Dataset $\mathcal{Y}$; pretrained encoder f_{enc}; graphette prior $\mathcal{W} = (W, \rho_n, f)$; batch size B; epochs $\mathcal{R}$; weights (λ_x, λ_e); learning rate η; FGW tradeoff α.
output Velocity Field (GNN Transformer) v_θ.

Initialize v_θ and initialize optimizer $\mathrm{AdamW}(\theta, \eta)$
for epoch $= 1$ to $\mathcal{R}$ **do**
  for all minibatch of target graphs $\mathcal{G}_1 = \{G_1^j\}_{j=1}^B \sim \mathcal{D}$ **do**
    Let $n_j \leftarrow |V(G_1^j)|$
    // Sample a batch of noise graphs from a graphette
    $\mathcal{G}_0 = \{G_0^i\}_{i=1}^B \sim \mathcal{W}$
    // Batch FGW-distances (Algorithm 5)
    $(D, T) \leftarrow \textsc{BatchFGW}(\mathcal{G}_0, \mathcal{G}_1, f_{enc}, \alpha)$
    // Hungarian matching on FGW costs to get matched graph pairs
    $\pi^\star \leftarrow \textsc{HungarianMatching}(D)$
    Coupled pairs $\mathcal{P} \leftarrow \{(G_0^i, G_1^{\pi^\star(i)})\}_{i=1}^B$
    // Batch gradient update for v_θ (Algorithm 6)
    $\theta \leftarrow \textsc{FlowMatchingUpdate}(\mathcal{P}, \theta)$
  end for
end for
return v_θ

Local Velocity Matching Loss. We enforce local flow correctness by matching predicted velocities to the rectified target transport. Let $\lambda_x, \lambda_e \geq 0$ be scalar weighting coefficients, and let $N_A = n^2$, $N_X = n\, d_x$, $N_F = n^2 d_f$ denote the number of entries in A, X, and F respectively. Each term below is thus a Mean Squared Error (MSE) over its tensor's entries:

$$\mathcal{L}_{\mathrm{vel}} = \frac{1}{N_A}\|v_A - \Delta A\|_F^2 + \lambda_x \frac{1}{N_X}\|v_X - \Delta X\|_F^2 + \lambda_e \frac{1}{N_F}\|v_F - \Delta F\|_F^2, \tag{7}$$

where v_A, v_X, and v_F denote the predicted velocity components for A, X and F, respectively. The targets $\Delta A = A_1 - A_0$, $\Delta X = X_1 - X_0$, and $\Delta F = F_1 - F_0$ correspond to the rectified transport directions induced by the linear interpolation between the prior graph and the target graph. This loss enforces correct local transport behavior at each intermediate state along the rectified flow path.

Endpoint Consistency Loss. While $\mathcal{L}_{\mathrm{vel}}$ constrains the instantaneous transport direction at a randomly sampled intermediate time t, we additionally supervise the one-step extrapolated endpoint implied by the rectified flow parameterization. Given A_t, X_t, and F_t at time t, a single-step rectified integration predicts the endpoint

$$\widehat{A}_1 = A_t + (1-t)v_A, \quad \widehat{X}_1 = X_t + (1-t)v_X, \quad \widehat{F}_1 = F_t + (1-t)v_F, \tag{8}$$

where $\widehat{A}_1, \widehat{X}_1$ and $\widehat{F}_1$ denote the predicted adjacency, node features, and edge features at $t = 1$. We penalize deviations from the target graph, again as a per-entry MSE using the same normalization constants N_A, N_X, N_F defined above, via

$$\mathcal{L}_{\mathrm{end}} = \frac{1}{N_A}\|\widehat{A}_1 - A_1\|_F^2 + \lambda_x \frac{1}{N_X}\|\widehat{X}_1 - X_1\|_F^2 + \lambda_e \frac{1}{N_F}\|\widehat{F}_1 - F_1\|_F^2. \tag{9}$$

As we show in App. E.3.1, $\mathcal{L}_{\mathrm{end}}$ is an algebraically induced, non-uniform temporal reweighting of the same velocity residual used in $\mathcal{L}_{\mathrm{vel}}$, with weight $(1-t)^2$: a fixed velocity residual induces a larger predicted endpoint error when incurred earlier in the trajectory (t closer to 0) than later (t closer to 1), since it is extrapolated over more of the remaining path. Combining $\mathcal{L}_{\mathrm{vel}}$ and $\beta_{\mathrm{end}}\mathcal{L}_{\mathrm{end}}$ therefore places greater emphasis on velocity residuals with a larger effect on the predicted terminal state, which remains useful under finite-step numerical integration, where errors in early-trajectory velocity predictions have outsized influence on the final generated graph.

Chemistry-aware Regularization. To promote chemically valid molecular graphs, we incorporate two lightweight regularization terms applied at the predicted endpoint. First, a *soft valence constraint* ($\mathcal{L}_{\mathrm{val}}$) penalizes violations of atom valence limits by discouraging over-bonding through a soft valence penalty. Second, an *atom-type marginal matching* ($\mathcal{L}_{\mathrm{atom}}$) term aligns the global distribution of predicted atom types with that of the target graph, preventing drift in node-type composition. Full formulations are provided in Appendix D.

Table 1: Graphette priors, as used in empirical benchmarks in Section 5.

Application	Graphette $\mathcal{W} = (W, \rho_n, f)$
Community graphs	$f = I$ (GEF 1), $\rho_n = 1$, and W is a mixture of stochastic block model graphons
Tree graphs	$f = h$ (GEF 2), $\rho_n = (\bar{W}n)^{-1} + \epsilon$ and $W(x, y) = 0.2$
Egonets	$f = R \circ S$ (GEF 3 and 4), $\rho_n = 1$ and $W(x, y) = \exp\left(-\frac{10}{3}(x + y)\right)$
Molecular graphs	$f = R$ (GEF 3), $\rho_n = (\bar{W}n)^{-1} + \epsilon$, where $\bar{W} = \int_0^1 \int_0^1 W(x, y)dxdy$ and $W(x, y) = 0.2$

Theoretical Properties of the Learning Objective. Theorem E.4 shows that, under the rectified parameterization, jointly driving $\mathcal{L}_{\text{vel}}$ and $\mathcal{L}_{\text{end}}$ to zero forces v_θ to recover the ideal constant transport field and guarantees exact endpoint reconstruction under Euler integration; this is consistent with $\mathcal{L}_{\text{end}}$ being a reweighted view of the velocity residual rather than an independent target, since driving that residual to zero at every t trivially drives both losses to zero together. Theorem E.5 separately quantifies how deviations from the ideal velocity field control the endpoint error under K-step Euler integration; in the rectified setting, the ideal velocity is constant, so no additional discretisation-error term arises from the Euler solver. For the computational complexity of these components, see Appendix F, and for runtime analysis, see Appendix J.7.

4 Structural Bias with Graphette Priors

Inspired by the star function in graphexes [66] and sparsified graphons we propose a generalized model that includes a graph edit function, which can edit graphs sampled from W_n or W. In this section $G = (V, E)$ refers to non-attributed graphs.

Definition 4.1. (**The graphette**) We define $\mathcal{W} = (W, \rho_n, f)$ where $W : [0, 1]^2 \to [0, 1]$ denotes a graphon, $\{\rho_i\}_i$ denotes a real-valued sequence with $0 \leq \rho_i \leq 1$ and f denotes a graph edit function. A graph is generated using graphette $\mathcal{W}$ by first sampling G' from $W_n := \rho_n W$ and then enacting the function f, i.e., $G' \sim \mathsf{G}(n, W, \rho_n)$ followed by $G \sim f(G')$.

For the computationally inclined reader, we provide a generative model description of graphettes and graphons in Appendix G. Our model has the flexibility to sample dense and sparse graphs. For sparse graphs, the graphette $\mathcal{W}$ is sparsified by $\rho_n \to 0$ (Def. A.8). As is common in other GNN papers [43, 67] when a sample from the sparsified graphon (Def. A.8) returns graphs with multiple components, we choose the largest connected component as the graph before applying the edits described below.

4.1 Rich and recurring subgraph motifs

We can choose a graph edit function f to suit our purpose. If a graphon W can generate the graphs we require, such as stochastic block model graphs, we can let f be the identity function, making no modifications to the graph. If we are modeling social networks, then f can add stars. Ring addition is useful to model molecular networks as they have ring structures such as benzene.

Motivated by these problem domains we use four graph edit functions (GEF) in our experiments: **GEF 1:** $f = I$ the identity function in instances where the graph does not need to be edited, **GEF 2:** $f = h$, a cycle deletion function when we know the graphs do not have cycles, **GEF 3:** $f = R$, a ring addition function when the graphs exhibit ring structures, and **GEF 4:** $f = S$, a star addition function for graphs with hub structures such as social networks.

Remark 4.2. Details of graph edit functions along with some characteristics of graphettes are explored in Appendix H. In particular, graphettes can recover graphon W and the sparsified graphon W_n, have the ability to generate graphs similar to those generated by graphexes and generate dense graphs from a family of graphons. Furthermore, the ring function GEF 3 can generate dense or sparse graphs depending on $\{\rho_i\}_i$ (see Lemmas in Appendix H).

We specify f manually per dataset family (Table 1); Sec. 5.4 discusses learned alternatives and quantifies sensitivity to graphette misspecification (see also App. K).

4.2 Star and ring functions are well-behaved

Graph homomorphisms are widely used to count subgraph motifs and have been used in GNNs [29].

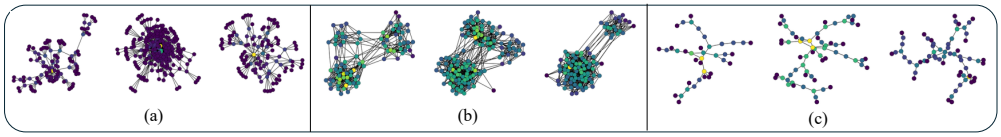

Figure 2: Flowette-generated synthetic graphs: Ego-small (a), SBM (b), and Tree (c).

Definition 4.3. Let G_1 and G_2 be graphs. A **graph homomorphism** from G_1 to G_2 is a map $f : V(G_1) \to V(G_2)$ such that if $uv \in E(G_1)$ then $f(u)f(v) \in E(G_2)$. (Maps edges to edges.) Let $\mathrm{Hom}(G_1, G_2)$ be the set of all such homomorphisms and let $\hom(G_1, G_2) = |\mathrm{Hom}(G_1, G_2)|$ be the number of homomorphisms. Then **homomorphism density** is defined as $t(G_1, G_2) = \frac{\hom(G_1,G_2)}{|V(G_2)|^{|V(G_1)|}}$.

Definition 4.4. (**Triangle-covered graphs**.) Let $G = (V, E)$ be a graph. Then G is triangle-covered if for every vertex $u \in V(G)$ there exist vertices $\{v, w\} \in V(G)$ such that edges $\{uv, uw, vw\} \in E(G)$. We denote the set of triangle-covered graphs by $\mathcal{F}$.

Roughly stated a graph is triangle-covered if every vertex belongs to a triangle. We show that homomorphisms of triangle-covered graphs are preserved by star and ring functions (GEF 3, 4).

Theorem 4.5. *Let $\mathcal{W} = (W, \rho_n, f)$ denote a graphette and let $G' \sim \mathsf{G}(n, W, \rho_n)$ and $G \sim f(G')$ where $f \in \{R, S\}$ (GEF 3, 4) with ring size $c > 3$ for R. Suppose $|V(G')| = n$ and $|V(G)| = n + m$. Then for any triangle-covered $F \in \mathcal{F}$, $\hom(F, G) = \hom(F, G')$ and $t(F, G) = \frac{n^{|V(F)|}}{(n+m)^{|V(F)|}} t(F, G')$.*

Theorems H.8 and H.9 supplement the above result by computing the effect of star and ring functions on vertex and edge homomorphisms and their densities.

5 Numerical Experiments

In this section, we evaluate the proposed flow matching framework on synthetic graph generation benchmarks and real-world molecular graph generation tasks. The synthetic benchmarks are designed to probe structural fidelity under controlled graph families, while the molecular benchmarks assess chemical validity, diversity, and novelty in realistic settings [2].

5.1 Graphette priors

Graphs generated from the $\mathcal{G}$ operator are used as priors in the flow matching framework. It is important to select an appropriate graphette prior that matches the graphs being modeled. Table 1 lists the graphette priors used for each application.

5.2 Experiments on Synthetic Graphs

We evaluate Flowette on three standard synthetic graph generation benchmarks: Tree [4], Stochastic Block Model (SBM) [51], and Ego-small graphs [32]. We adopt the same training, validation, and test splits as in [55]. Generation quality is assessed using seven different metrics, described in more detail in Appendix J.2. We exclude chemistry-specific regularization terms for these synthetic data, by setting $\beta_{\text{val}} = 0$ and $\beta_{\text{atom}} = 0$. We used $L = 3$ attention layers for Tree and Ego-small datasets, and $L = 4$ for the SBM dataset. Other hyperparameter settings are provided in Appendix J.3.

We summarise results on synthetic datasets in Tables 5–7. Across SBM, Tree, and Ego-small, Flowette achieves the strongest performance on higher-order structural metrics (e.g., Orbit) and generative quality measures (e.g., Validity, Uniqueness, Novelty, and V.U.N.), while remaining competitive on Degree and Clustering. This reflects the model's ability to capture global structure and motif-level patterns (e.g., community structure in SBM, acyclicity in Tree, and hub-centric motifs in Ego-small), with minor deviations in lower-order statistics (e.g., Deg. and Clus.) that do not affect overall graph fidelity. Tables of results against baseline models are shown in Appendix J.4. Figure 2 presents representative synthetic graphs generated by Flowette on the Ego-small, SBM, and Tree datasets. Additional samples in Appendix L further demonstrate the validity of the generative model.

[2] Code: https://github.com/wokas36/Flowette

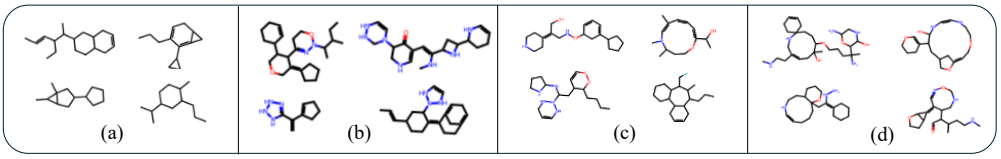

Figure 3: Flowette-generated molecules: QM9 (a), ZINC250K (b), MOSES (c) and Guacamol (d).

Table 2: Generation performance on QM9 and ZINC250K molecular graphs.

	QM9			ZINC250K		
Model	Valid ↑	Unique ↑	NSPDK ↓	Valid ↑	Unique ↑	NSPDK ↓
EDP-GNN [53]	47.52	99.25	0.0046	82.97	99.79	0.0485
GraphARM [37]	90.25	95.65	0.002	88.23	99.46	0.055
DiGress [67]	98.19	96.20	0.0003	94.99	99.97	0.0021
GDSS [32]	95.72	98.46	0.0033	97.01	99.64	0.0195
GDSS + TF [32]	99.68	-	0.0024	96.04	-	0.0326
GruM [31]	99.69	96.90	0.0002	98.65	99.97	0.0015
GraphBFN [60]	99.73	-	0.0002	99.22	-	0.0013
SID [5]	99.67	95.66	**0.0001**	99.50	99.84	0.0021
DeFoG [55]	99.30	96.30	-	99.22	99.99	0.0008
GBD [47]	**99.88**	-	0.0002	97.87	-	0.0018
Flowette (Ours)	99.81 $\pm$ 0.09	**99.30** $\pm$ 0.05	0.0003 $\pm$ 0.0002	**99.90** $\pm$ 0.10	**100.00** $\pm$ 0.0	**0.0006** $\pm$ 0.0004

5.3 Experiments on Molecular Graphs

We further evaluate our framework on four standard molecular graph generation benchmarks: QM9 [68], ZINC250K [62], Guacamol [8], and MOSES [54], which together span small molecules and larger, drug-like compounds. Experiments on QM9 use the conventional dataset splits and evaluation criteria employed in recent molecular generation work [67, 55]. For MOSES and Guacamol, we follow the respective benchmark protocols for training and assessment [54, 8]. Details of the metrics are presented in Appendix J.5. As can be seen in Table 2 and 3, our approach achieves competitive performance across all metrics, with state-of-the-art performance on several metrics while remaining comparable to strong baselines on others. Figure 3 shows representative molecular graphs generated by Flowette on QM9, ZINC250K, MOSES, and Guacamol datasets. We used $L = 4$ attention layers for Guacamol and MOSES, and $L = 5$ for QM9 and ZINC250K. Further details of our experimental protocol are provided in Appendix J.6.

5.4 Sensitivity to Graphette Edit-Function Misspecification

As noted in Sec. 4, f is specified manually per dataset family (Table 1) because the graphette is intended only as a coarse structural bias on the flow's source distribution, not as the final generative model: it supplies domain-level topology – rings, hubs, or acyclicity – while the velocity field v_θ learns the remaining target-specific structure. This also lets us isolate the benefit of structural source bias without introducing a second learned generator. The graphette's components need not remain manually specified in general: the base graphon W could be estimated directly from data using implicit graphon-learning methods such as Implicit Graphon Neural Representation [69] or Scalable Implicit Graphon Learning [3]; the sparsification sequence ρ_n could be fitted to match empirical edge-density scaling in n; and the edit function f could be selected or learned from motif statistics or validation performance. We use simple, interpretable priors in this work to isolate the contribution of structural source bias; the remainder of this section quantifies sensitivity to the choice of f.

The ablation in App. J.9.3 (graphette prior vs. a standard graphon) removes structural bias entirely and therefore represents an extreme mismatch. To more precisely quantify sensitivity to *which* motif the graphette encodes, we hold every other component of Flowette fixed – FGW coupling, the velocity-field architecture, and the training objective – and vary only the edit function f used in the molecular graphette (Table 1). The results are shown in Table 4. The intended prior for molecular graphs uses ring addition R, reflecting the prevalence of cyclic structure in molecules. We compare it against two increasingly incompatible choices: star addition S, a structurally unrelated motif, and cycle deletion h, which is directly contradictory since it actively removes the ring structure the target domain requires. Validity degrades consistently as the prior becomes less compatible with the target domain: on QM9, replacing R with S reduces validity from 99.81% to 70.50%, and replacing it with the contradictory h reduces validity further to 53.70%, with analogous trends on ZINC250K and

Table 3: Generation performance on Guacamol and MOSES molecular graphs.

	Guacamol			MOSES			
Model	Valid ↑	V.U.N ↑	KL div. ↑	Unique ↑	Novelty ↑	SNN ↑	Scaff. ↑
DiGress [67]	85.2	85.1	92.9	**100.0**	95.0	0.52	14.8
DisCo [71]	86.6	86.5	92.6	**100.0**	97.7	0.50	15.1
Cometh [59]	98.9	97.6	96.7	99.9	92.6	0.54	**16.0**
GraphBFN [60]	-	-	-	99.8	89.0	**0.59**	10.0
G2PT [14]	95.3	94.8	95.6	**100.0**	79.4	0.55	2.9
GEEL [28]	88.2	77.1	93.1	**100.0**	81.1	0.52	3.6
DeFoG [55]	**99.0**	97.9	97.7	99.9	92.1	0.55	14.4
Flowette (Ours)	98.6	**98.0**	**97.9**	99.9	**98.1**	0.58	15.3

Table 4: Sensitivity of Flowette to graphette edit-function misspecification on QM9 and ZINC250K. All components other than f are held fixed.

Dataset	Edit function	Prior variant	Valid	Unique	NSPDK
QM9	R (Ring addition)	Intended prior	$\mathbf{99.81}$ $\pm$ 0.09	$\mathbf{99.30}$ $\pm$ 0.05	$\mathbf{0.0003}$ $\pm$ 0.0002
	S (Star addition)	Incorrect motif	70.50 $\pm$ 0.06	69.30 $\pm$ 0.06	0.0160 $\pm$ 0.0008
	h (Cycle deletion)	Contradictory motif	53.70 $\pm$ 0.04	54.45 $\pm$ 0.06	0.0540 $\pm$ 0.0004
ZINC250K	R (Ring addition)	Intended prior	$\mathbf{99.90}$ $\pm$ 0.10	$\mathbf{100.00}$ $\pm$ 0.0	$\mathbf{0.0006}$ $\pm$ 0.0004
	S (Star addition)	Incorrect motif	73.40 $\pm$ 0.06	75.30 $\pm$ 0.06	0.1105 $\pm$ 0.0006
	h (Cycle deletion)	Contradictory motif	51.58 $\pm$ 0.04	50.65 $\pm$ 0.05	0.2050 $\pm$ 0.0007

substantial increases in NSPDK under both misspecified priors. Flowette therefore benefits from – and is sensitive to – a coarse but structurally appropriate source prior, rather than being invariant to the choice of graphette edit function. This is consistent with, but a finer-grained probe than, the graphette-vs-graphon ablation of App. J.9.3: here all variants retain a graphette-structured source, differing only in whether the encoded motif matches the target domain.

5.5 Ablation Analysis of Regularization Terms.

Table 10 isolates the components of the Flowette objective (Eq. (6)) on QM9 and ZINC250K. Removing $\mathcal{L}_{\text{end}}$ or $\mathcal{L}_{\text{val}}$ substantially reduces validity, whereas removing $\mathcal{L}_{\text{atom}}$ has a milder effect on validity but consistently reduces uniqueness and novelty, confirming that chemical feasibility does not emerge automatically from generic flow-matching supervision – without indicating that the backbone itself is molecule-specific, since the same backbone achieves state-of-the-art on our synthetic benchmarks on several metrics (Sec. 5.2) with $\beta_{\text{val}} = \beta_{\text{atom}} = 0$. Full discussion, including why $\mathcal{L}_{\text{vel}}$ alone is insufficient on molecular graphs and how the endpoint-aware temporal reweighting places greater emphasis on residuals with a longer remaining extrapolation horizon, is provided in App. J.9.1; further ablations on the encoder sensitivity, graphette prior, and FGW coupling are in App. J.9.2, App. J.9.3, and App. J.9.4, respectively.

Observed limitations. Several aspects of Flowette offer avenues for further improvement, spanning coupling scalability, automated graphette specification, domain-general endpoint regularisation, and tighter train/generation integration; we discuss these limitations in App. K.

6 Conclusion

We propose Flowette, a generative model for graphs with recurring sub-graph motifs, built around three components: (1) structural alignment of supervision pairs via Fused Gromov-Wasserstein optimal transport; (2) a training objective that combines local velocity matching with an endpoint-aware reweighting of the same residual, together with optional chemistry-aware terms for molecule generation; (3) source distributions that encode structural priors via *graphettes*, a new mathematical object that extends graphons through controlled sub-graph edits (ring or star injection, cycle removal). The velocity field is learned by a GNN-based transformer. Flowette achieves competitive or state-of-the-art performance on several metrics across multiple synthetic and molecular benchmarks, and our ablations confirm that each of the three components contributes independently to performance. We hope each of these components; structural alignment, endpoint-aware regularisation, and graphette priors, will be useful in their own right for future work in graph generation.

References

[1] D. J. Aldous. Representations for partially exchangeable arrays of random variables. *Journal of Multivariate Analysis*, 11(4):581–598, 1981.

[2] J. Austin, D. D. Johnson, J. Ho, D. Tarlow, and R. van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems 34*, 2021.

[3] A. Azizpour, N. Zilberstein, and S. Segarra. Scalable implicit graphon learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2025.

[4] A. Bergmeister, K. Martinkus, N. Perraudin, and R. Wattenhofer. Efficient and scalable graph generation through iterative local expansion. In *International Conference on Learning Representations*, 2023.

[5] Y. Boget. Simple and critical iterative denoising: A recasting of discrete diffusion in graph generation. In *International Conference on Machine Learning*, 2025.

[6] C. Borgs, J. Chayes, L. Lovász, V. Sós, and K. Vesztergombi. Limits of randomly grown graph sequences. *European Journal of Combinatorics*, 32(7):985–999, 2011.

[7] C. Borgs, J. T. Chayes, L. Lovász, V. T. Sós, and K. Vesztergombi. Convergent sequences of dense graphs i: Subgraph frequencies, metric properties and testing. *Advances in Mathematics*, 219(6):1801–1851, 2008.

[8] N. Brown, M. Fiscato, M. H. Segler, and A. C. Vaucher. Guacamol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3):1096–1108, 2019.

[9] A. Campbell, J. Benton, V. De Bortoli, T. Rainforth, G. Deligiannidis, and A. Doucet. A continuous time framework for discrete denoising models. In *Advances in Neural Information Processing Systems 35*, 2022.

[10] A. Campbell, J. Yim, R. Barzilay, T. Rainforth, and T. Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. In *International Conference on Machine Learning*, 2024.

[11] F. Caron and E. B. Fox. Sparse graphs using exchangeable random measures. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 79(5):1295–1366, 2017.

[12] H. Chen, N. Zilberstein, and S. Segarra. Prior-informed flow matching for graph reconstruction. *arXiv preprint arXiv:2601.22107*, 2026.

[13] X. Chen, J. He, X. Han, and L. Liu. Efficient and degree-guided graph generation via discrete diffusion modeling. In *International Conference on Machine Learning*, pages 4585–4610. PMLR, 2023.

[14] X. Chen, Y. Wang, J. He, Y. Du, S. Hassoun, X. Xu, and L. Liu. Graph generative pre-trained transformer. In *International Conference on Machine Learning*, pages 9176–9197. PMLR, 2025.

[15] H. Dai, A. Nazi, Y. Li, B. Dai, and D. Schuurmans. Scalable deep generative modeling for sparse graphs. In *International conference on machine learning*, pages 2302–2312. PMLR, 2020.

[16] A. O. Davies, N. S. Ajmeri, et al. Size matters: Large graph generation with higgs. *arXiv preprint arXiv:2306.11412*, 2023.

[17] N. De Cao and T. Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

[18] N. L. Diamant, A. M. Tseng, K. V. Chuang, T. Biancalani, and G. Scalia. Improving graph generation by restricting graph bandwidth. In *International Conference on Machine Learning*, pages 7939–7959. PMLR, 2023.

[19] F. Eijkelboom, G. Bartosh, C. Andersson Naesseth, M. Welling, and J.-W. van de Meent. Variational flow matching for graph generation. *Advances in Neural Information Processing Systems*, 37:11735–11764, 2024.

[20] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[21] A. Frieze and R. Kannan. Quick approximation to matrices and applications. *Combinatorica*, 19(2):175–220, 1999.

[22] I. Gat, T. Remez, N. Shaul, F. Kreuk, R. T. Chen, G. Synnaeve, Y. Adi, and Y. Lipman. Discrete flow matching. *Advances in Neural Information Processing Systems*, 37:133345–133385, 2024.

[23] N. Goyal, H. V. Jain, and S. Ranu. Graphgen: A scalable approach to domain-agnostic labeled graph generation. In *Proceedings of the web conference 2020*, pages 1253–1263, 2020.

[24] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 2012.

[25] K. K. Haefeli, K. Martinkus, N. Perraudin, and R. Wattenhofer. Diffusion models for graphs benefit from discrete state spaces. *NeurIPS New Frontiers in Graph Learning Workshop (NeurIPS GLFrontiers)*, 2022.

[26] X. Han, X. Chen, F. J. Ruiz, and L.-P. Liu. Fitting autoregressive graph generative models through maximum likelihood estimation. *Journal of Machine Learning Research*, 2023.

[27] X. Han, Z. Jiang, N. Liu, and X. Hu. G-mixup: Graph data augmentation for graph classification. In *International conference on machine learning*, pages 8230–8248. PMLR, 2022.

[28] Y. Jang, S. Lee, and S. Ahn. A simple and scalable representation for graph generation. In *International Conference on Learning Representations*, 2024.

[29] E. Jin, M. Bronstein, İ. İ. Ceylan, and M. Lanzinger. Homomorphism counts for graph neural networks: all about that basis. In *Proceedings of the 41st International Conference on Machine Learning*, pages 22075–22098, 2024.

[30] W. Jin, R. Barzilay, and T. Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pages 2323–2332. PMLR, 2018.

[31] J. Jo, D. Kim, and S. J. Hwang. Graph generation with diffusion mixture. In *International Conference on Machine Learning*, 2024.

[32] J. Jo, S. Lee, and S. J. Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International conference on machine learning*, pages 10362–10383. PMLR, 2022.

[33] M. Karami. Higen: Hierarchical graph generative networks. *arXiv preprint arXiv:2305.19337*, 2023.

[34] N. Keriven, A. Bietti, and S. Vaiter. Convergence and stability of graph convolutional networks on large random graphs. *Advances in Neural Information Processing Systems*, 33:21512–21523, 2020.

[35] T. N. Kipf and M. Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[36] O. Klopp, A. B. Tsybakov, and V. Nicolas. Oracle inequalities for network models and sparse graphon estimation. *The Annals of Statistics*, 45(1):316–354, 2017.

[37] L. Kong, J. Cui, H. Sun, Y. Zhuang, B. A. Prakash, and C. Zhang. Autoregressive diffusion model for graph generation. In *International conference on machine learning*, pages 17391–17408. PMLR, 2023.

[38] H. W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[39] R. Liao, Y. Li, Y. Song, S. Wang, W. Hamilton, D. K. Duvenaud, R. Urtasun, and R. Zemel. Efficient graph generation with graph recurrent attention networks. *Advances in neural information processing systems*, 32, 2019.

[40] S. Limnios, P. Selvaraj, M. Cucuringu, C. Maple, G. Reinert, and A. Elliott. Sagess: Sampling graph denoising diffusion model for scalable graph generation. *arXiv preprint arXiv:2306.16827*, 2023.

[41] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

[42] P. Lippe and E. Gavves. Categorical normalizing flows via continuous transformations. In *International Conference on Learning Representations*, 2021.

[43] G. Liu, J. Xu, T. Luo, and M. Jiang. Graph diffusion transformers for multi-conditional molecular generation. *Advances in Neural Information Processing Systems*, 37:8065–8092, 2024.

[44] J. Liu, A. Kumar, J. Ba, J. Kiros, and K. Swersky. Graph normalizing flows. *Advances in Neural Information Processing Systems*, 32, 2019.

[45] M. Liu, K. Yan, B. Oztekin, and S. Ji. Graphebm: Molecular graph generation with energy-based models. In *ICLR Workshop on Energy Based Models*, 2021.

[46] X. Liu, C. Gong, and Q. Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023.

[47] X. Liu, Y. He, B. Chen, and M. Zhou. Advancing graph generation through beta diffusion. In *International Conference on Learning Representations*, 2025.

[48] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[49] N. Ma, M. Goldstein, M. S. Albergo, N. M. Boffi, E. Vanden-Eijnden, and S. Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, 2024.

[50] O. Mahmood, E. Mansimov, R. Bonneau, and K. Cho. Masked graph modeling for molecule generation. *Nature communications*, 2021.

[51] K. Martinkus, A. Loukas, N. Perraudin, and R. Wattenhofer. Spectre: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *International Conference on Machine Learning*, pages 15159–15179. PMLR, 2022.

[52] R. Mercado, T. Rastemo, E. Lindelöf, G. Klambauer, O. Engkvist, H. Chen, and E. J. Bjerrum. Graph networks for molecular design. *Machine Learning: Science and Technology*, 2(2):025023, 2021.

[53] C. Niu, Y. Song, J. Song, S. Zhao, A. Grover, and S. Ermon. Permutation invariant graph generation via score-based generative modeling. In *International conference on artificial intelligence and statistics*, pages 4474–4484. PMLR, 2020.

[54] D. Polykovskiy, A. Zhebrak, B. Sanchez-Lengeling, S. Golovanov, O. Tatanov, S. Belyaev, R. Kurbanov, A. Artamonov, V. Aladinskiy, M. Veselov, et al. Molecular sets (moses): a benchmarking platform for molecular generation models. *Frontiers in pharmacology*, 11:565644, 2020.

[55] Y. Qin, M. Madeira, D. Thanou, and P. Frossard. Defog: Discrete flow matching for graph generation. In *International Conference on Machine Learning (ICML)*, 2025.

[56] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 2011.

[57] C. Shi, M. Xu, Z. Zhu, W. Zhang, M. Zhang, and J. Tang. Graphaf: A flow-based autoregressive model for molecular graph generation. In *International Conference on Learning Representations*, 2020.

[58] H. Shirzad, H. Hajimirsadeghi, A. H. Abdi, and G. Mori. Td-gen: graph generation using tree decomposition. In *International Conference on Artificial Intelligence and Statistics*, pages 5518–5537. PMLR, 2022.

[59] A. Siraudin, F. D. Malliaros, and C. Morris. Cometh: A continuous-time discrete-state graph diffusion model. *Transactions on Machine Learning Research*, 2024.

[60] Y. Song, J. Shi, J. Gong, M. Xu, S. Ermon, H. Zhou, and W.-Y. Ma. Smooth interpolation for improved discrete graph generative models. In *International Conference on Machine Learning*, 2025.

[61] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[62] T. Sterling and J. J. Irwin. Zinc 15–ligand discovery for everyone. *Journal of chemical information and modeling*, 55(11):2324–2337, 2015.

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

[64] V. Titouan, N. Courty, R. Tavenard, and R. Flamary. Optimal transport for structured data with application on graphs. In *International Conference on Machine Learning (ICML)*, 2019.

[65] V. Veitch and D. M. Roy. The class of random graphs arising from exchangeable random measures. *arXiv preprint arXiv:1512.03099*, 2015.

[66] V. Veitch and D. M. Roy. Sampling and estimation for (sparse) exchangeable graphs. *The Annals of Statistics*, 47(6):3274–3299, 2019.

[67] C. Vignac, I. Krawczuk, A. Siraudin, B. Wang, V. Cevher, and P. Frossard. DiGress: Discrete Denoising diffusion for graph generation. In *The Eleventh International Conference on Learning Representations*, 2023.

[68] Z. Wu, B. Ramsundar, E. N. Feinberg, J. Gomes, C. Geniesse, A. S. Pappu, K. Leswing, and V. Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

[69] X. Xia, G. Mishne, and Y. Wang. Implicit graphon neural representation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 10619–10634. PMLR, 2023.

[70] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

[71] Z. Xu, R. Qiu, Y. Chen, H. Chen, X. Fan, M. Pan, Z. Zeng, M. Das, and H. Tong. Discrete-state continuous-time diffusion for graph generation. *Advances in Neural Information Processing Systems*, 37:79704–79740, 2024.

[72] K. Yi, B. Zhou, Y. Shen, P. Liò, and Y. Wang. Graph denoising diffusion for inverse protein folding. *Advances in Neural Information Processing Systems*, 36:10238–10257, 2023.

[73] J. You, R. Ying, X. Ren, W. Hamilton, and J. Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pages 5708–5717. PMLR, 2018.

[74] C. Zang and F. Wang. Moflow: an invertible flow model for generating molecular graphs. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020.

[75] Y. Zhu, Y. Du, Y. Wang, Y. Xu, J. Zhang, Q. Liu, and S. Wu. A survey on deep graph generation: Methods and applications. In *Learning on Graphs Conference*, pages 47–1. PMLR, 2022.

A Background on Graphons

Graphons are symmetric, measurable functions defined on a unit square denoted by $W : [0,1]^2 \to [0,1]$. Graphons are graph limits, i.e., by mapping the adjacency matrix to the unit square and taking the limit as the number of nodes go to infinity, we can obtain a graphon for converging graph sequences. When the adjacency matrix is mapped to the unit square, we obtain the empirical graphon, defined below.

A.1 Graphons are graph limits

Definition A.1. Given a graph G with n vertices labeled $\{1, \ldots, n\}$, we define its **empirical graphon** $W_G : [0,1]^2 \to [0,1]$ as follows: We split the interval $[0,1]$ into n equal intervals $\{I_1, I_2, \ldots, I_n\}$ (first one closed, all others half open) and for $x \in I_i, y \in I_j$ define

$$W_G(x,y) = \begin{cases} 1 & \text{if} \quad ij \in E(G) \\ 0 & \text{otherwise}\,, \end{cases}$$

where $E(G)$ denotes the edges of G. The empirical graphon replaces the adjacency matrix with a unit square and the (i,j)th entry of the adjacency matrix is replaced with a square of size $(1/n) \times (1/n)$.

Figure 4 shows the empirical graphons for 3 graphs generated from the Erdős–Rényi $G(n,p)$ model where G has n vertices for $n \in \{50, 100, 500\}$ with edge probability $p = 0.2$. This graph sequence converges to $W = 0.2$.

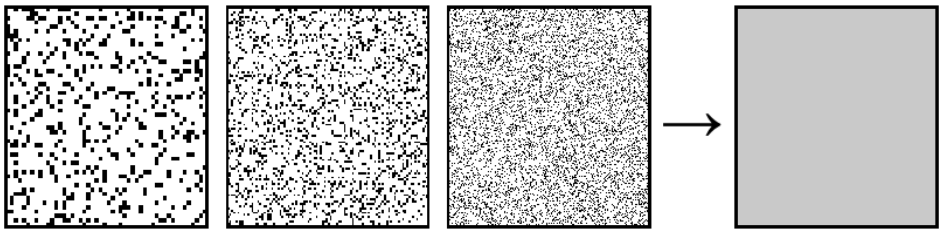

Figure 4: Three $G(n,p)$ graphs for $n \in \{50, 100, 500\}$ with edge probability $p = 0.2$ converging to $W = 0.2$.

Graph convergence is defined in terms of graph homomorphism density and the cut-metric [7]. Both these notions are equivalent [6]. We define the cut norm and the cut metric below.

Definition A.2. The **cut norm** of graphon W [21, 7] is defined as

$$\|W\|_\square = \sup_{S,T} \left| \int_{S \times T} W(x,y)\, dxdy \right| ,$$

where the supremum is taken over all measurable sets S and T of $[0,1]$.

Definition A.3. Given two graphons W_1 and W_2 the **cut metric** [7] is defined as

$$\delta_\square(W_1, W_2) = \inf_\varphi \|W_1 - W_2^\varphi\|_\square \,,$$

where the infimum is taken over all measure preserving bijections $\varphi : [0,1] \to [0,1]$.

The role of φ is to account for graph isomorphisms. Two isomorphic graphs G and H may have different node labels making $\|W_G - W_H\|_\square$ non-zero. However, for these two graphs $\inf_\varphi \|W_G - W_H^\varphi\|_\square$ is zero as φ accounts for bijections.

As shown in [6] a graph sequence $\{G_n\}_n$ is convergent if and only if it is convergent in the cut metric.

Theorem A.4 (Graph convergence [6])**.** *A sequence of graphs $\{G_n\}_n$ is convergent if and only if it is Cauchy in the $\delta_\square$ distance. The sequence $\{G_n\}_n$ converges to W if and only if $\delta_\square(W_{G_n}, W) \to 0$. Furthermore, if this is the case, and $|V(G_n)| \to \infty$, then there is a way to label the nodes of the graphs G_n such that $\|W_{G_n} - W\|_\square \to 0$.*

A.2 Graphons are graph generative models

Graphons can be used to generate new graphs. Given a graphon W we can sample a graph with n nodes as follows:

Definition A.5. Uniformly pick $x_1, x_2, \ldots, x_n$ from $[0, 1]$. A **W-random graph** $\mathsf{G}(n, W)$ has the vertex set $1, 2, \ldots, n$ and vertices i and j are connected with probability $W(x_i, x_j)$. That is, for each node pair i and j we add an edge $A_{ij} \sim \text{Bernoulli}(W(x_i, x_j))$.

Figure 5 shows two graphons and graphs generated from them for different values of n. The graphon on the top row generates ring type graphs and the graphon on the bottom row generates stochastic block model type graphs.

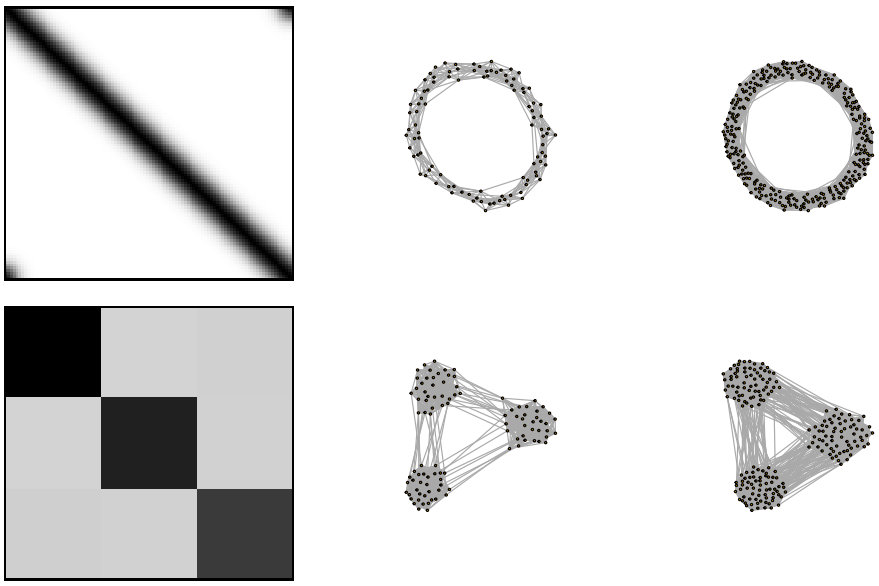

Figure 5: Two graphons on the left and W-random graphs $\mathsf{G}(n, W)$ for different n on the right.

Next we introduce dense and sparse graph sequences.

Definition A.6 (Dense graph sequences). A sequence of graphs $\{G_n\}_n$ is dense if the number of edges m grow quadratically with the number of nodes n, i.e., $m \in \Theta(n^2)$.

Definition A.7 (Sparse graph sequences). A sequence of graphs $\{G_n\}_n$ is sparse if the number of edges m grow sub-quadratically with the number of nodes n, i.e., $m \in o(n^2)$.

The limitation of graphons is that as a consequence of the Aldous-Hoover theorem [1], graphs generated from graphons are dense or empty. However, most real-world graphs such as social networks and molecular graphs are sparse. As such, many graphon-based extensions are proposed to mitigate this limitation.

A.3 Graphon extensions for sparse graphs

We discuss two sparse graph graphon extensions: graphon sparsification [36], and graphexes [65].

Definition A.8. (**Sparsified graphon**.) Let $W : [0, 1]^2 \to [0, 1]$ denote a graphon and $\{\rho_i\}_i = (\rho_1, \rho_2, \ldots, \rho_n, \ldots)$ denote a real-valued sequence converging to zero such that $0 \leq \rho_i \leq 1$. Then the graphon $W_n := \rho_n W$ is called the sparsified graphon. We denote a graph G with n nodes sampled from W_n by $G \sim \mathsf{G}(n, W, \rho_n)$.

Note that $\rho_n W \to 0$ with n. As such, if G_n is sampled from $\rho_n W$ then the sequence $\{G_n\}_n$ is sparse. Figure 6 shows a stochastic block model graphon W and two sparsified versions $\rho_n W$ for $\rho_n \in \{0.1, 0.05\}$ along with graphs sampled from these three graphons for $n \in \{100, 200, 300\}$. We see the block structure is diluted for smaller values of ρ_n both in the graphon and in the sampled graph. In addition, the number of isolated nodes increase as ρ_n decreases.

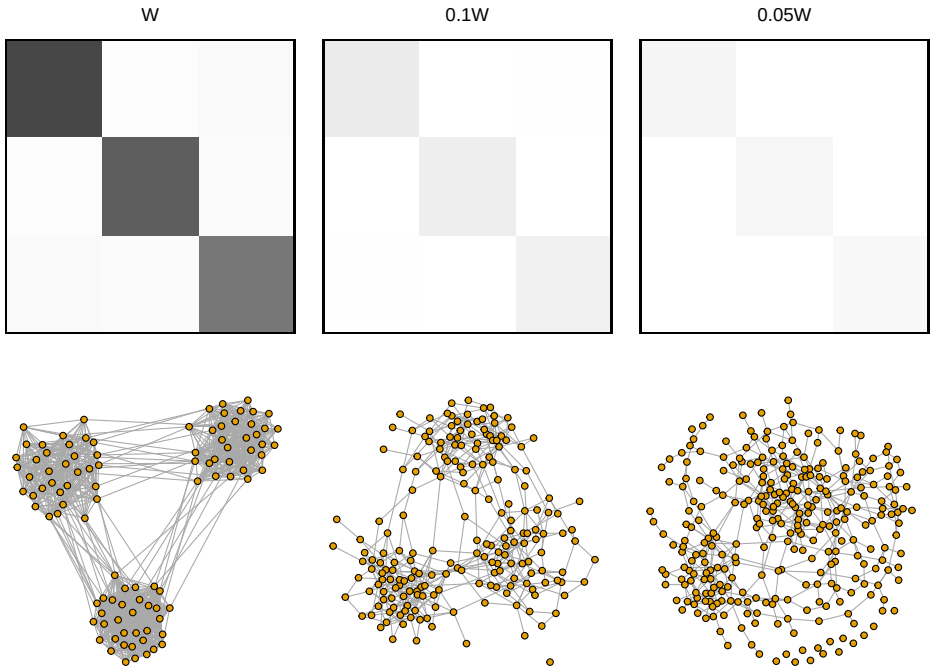

Figure 6: Graphon sparsification where $W_n = \rho_n W$. Top row shows the original graphon W and two instances of $\rho_n W$. The bottom row shows 3 graphs sampled from the graphon directly above.

While graphon sparsification achieves sparsity, it has limitations. The main limitation is that it doesn't create large hub structures prevalent in social networks. Graphexes [65] have the capacity to address this issue.

Graphexes. Graphexes were originally introduced by Veitch and Roy [65] as a triple $\mathcal{W} = (W, S, I)$ where

- $W : \mathbb{R}_+^2 \to [0, 1]$ is a symmetric measurable function (graphon component),
- $S : \mathbb{R}_+ \to \mathbb{R}_+$ is an integrable function (star component),
- $I \geq 0$ is a constant (dust/isolated edges component).

The main difference of graphexes from the original graphon framework is that motivated from the work by Caron and Fox [11] they sample graphs via a Poisson process construction and use Kallenberg exchangeability in their regularity framework to account for the exchangeability of vertices that are modeled as reals ($\mathbb{R}^+$) instead of integers. The Poisson process construction coupled with a graphon W defined on $\mathbb{R}_+^2$ allows them to generate sparse graphs. Having a separate star function gives the capability to add stars akin to social networks.

Our proposed structural prior, *graphettes*, takes inspiration from these two sparse graph models. We keep the graphon defined on the unit square, as this simplifies computations, and employ a sparsified graphon in our construction. To account for structures such as rings or stars that are not generated by the sparsified graphon W_n, we introduce a graph edit function f, similar to the star function S in graphexes. However, the graph edit function f has broader functionality than the star function, as it can modify the graph in multiple ways, for example, by adding rings or deleting edges. As a result, graphettes allow us to generate a diverse range of graphs, from molecular structures to social networks.

B Historical Background and Related Work on Flow Matching

B.1 Graph generative models

Graph generative modeling aims to learn distributions over graphs that capture both local connectivity patterns and global structural properties, with applications in molecular design, combinatorial optimisation, and biological networks [52, 63, 72]. Existing methods can be broadly grouped into autoregressive and one-shot paradigms. Autoregressive models sequentially grow a graph by inserting nodes and edges according to a chosen ordering [73, 39, 50, 23]. This sequential view allows fine-grained control and the injection of domain-specific constraints (e.g. valency checks in molecular generation), but comes at a cost: models must either learn an effective node/edge ordering or commit to a hand-crafted one, which enlarges the hypothesis space and breaks permutation invariance [37, 26]. As a result, autoregressive approaches tend to be computationally expensive, difficult to parallelise, and brittle on graphs with complex long-range dependencies.

To mitigate these issues, a large body of work has explored one-shot generation, where all graph variables are predicted in a single pass. One-shot models instantiate graph-adapted versions of VAEs, GANs, and normalising flows [35, 17, 44]. While they restore permutation equivariance/invariance and avoid learning an ordering, purely one-shot models often underfit rich higher-order structure unless equipped with carefully engineered architectures or hierarchical decompositions [15, 58, 33, 16].

Graph diffusion models. Diffusion-based generative models have recently become the dominant one-shot paradigm for graphs. Early work adapted continuous diffusion frameworks to graph-structured data by treating adjacency and features as continuous variables [53, 32], which simplifies optimisation but departs from the discrete nature of graph topology. Subsequent methods introduced discrete-state diffusion over node and edge categories using discrete-time Markov chains [2, 67, 25], achieving strong performance on benchmark graph and molecular datasets. However, these models tie the sampling trajectory to the training schedule: the number of reverse steps must match the (fixed) number of forward noising steps, and sampling cost is therefore tightly coupled to training hyperparameters.

Continuous-time discrete diffusion models based on Continuous Time Markov Chains (CTMCs) address this rigidity by decoupling the sampling grid from the training objective [9, 71]. DisCo [71], for instance, defines a discrete-state continuous-time diffusion on graphs and inherits the permutation-equivariant/invariant properties of DiGress while enabling flexible numerical integration and theoretical control of approximation error [71]. Other variants exploit degree information or hierarchical decompositions to improve scalability and sample quality [13, 4, 40]. Despite these advances, diffusion-based graph models still rely on carefully designed rate matrices or noise schedules, and their local node/edge updates make it difficult to preserve global topology and higher-order motifs without additional inductive biases or auxiliary constraints.

Graph flow matching models. Flow matching (FM) offers an efficient alternative to diffusion-based training for continuous-time generative models. Rather than learning to reverse a fixed stochastic process, FM directly regresses a time-dependent vector field whose trajectories transport a simple source distribution to a complex target distribution along a chosen probability path [41]. Rectified-flow variants further simplify training by adopting linear interpolants between data and noise and regressing the associated constant conditional velocity field [46], leading to state-of-the-art results in high-dimensional image and sequence generation [20, 49].

Extending FM to discrete state spaces has given rise to Discrete Flow Matching (DFM), where probability paths are defined by time-inhomogeneous CTMCs and the model parameterises transition rates rather than continuous velocities [10, 22]. Compared to discrete diffusion, DFM replaces hand-designed forward rate matrices with learnable dynamics and decouples the choice of interpolation path from the training objective. Building on this, DeFoG introduces the first DFM-based framework for graph generation, combining CTMC-based flows with a graph transformer backbone and a training–sampling decoupling that improves flexibility over purely diffusion-based methods [55].

In parallel, some works have explored continuous FM and flow-based formulations for graphs by relaxing discrete structures into continuous spaces. Graph normalising flows and related models learn invertible transports over continuous embeddings of adjacency and attributes [44], while variational flow matching for graph generation embeds graphs into continuous latent spaces and trains vector

fields via a variational objective [19]. These approaches demonstrate that FM-style training can be competitive to diffusion-based graph generators in terms of efficiency and flexibility. However, they typically rely on implicit couplings between source and target graphs (e.g. through shared encoders or attention over batched graphs) and do not explicitly enforce structural alignment under permutation and size variability. Consequently, velocity supervision can become inconsistent across permutations or mismatched graph components, making it difficult to guarantee stable preservation of global topology and higher-order motifs during integration. In particular, whereas DiGress [67], DeFoG [55], and VFM [19] predict categorical outputs or induce their vector field from a categorical endpoint, our GNN transformer directly integrates continuous, topology-conditioned velocities with persistent edge states (App. C.2).

Our work situates itself in this emerging line of flow-matching graph generators, but departs from prior art in two key ways: (1) we adopt a continuous FM framework over relaxed adjacency and node features while explicitly aligning noise and data graphs via structure-aware matching, and (2) we incorporate principled structural priors into both the source distribution and the training objective to capture local and global structural properties. This design yields coherent, permutation-invariant velocity supervision across variable-sized graphs and enables Flowette to learn stable continuous-time graph transformations that preserve rich combinatorial structure beyond what is typically captured by existing diffusion- or flow-based models.

B.2 Flow matching training

FM provides a framework for learning continuous-time generative models by directly regressing a time-dependent velocity field that transports samples from a simple source distribution to a target data distribution along a prescribed probability path [41]. Unlike diffusion-based models, FM does not require simulating or inverting a stochastic corruption process, and avoids likelihood evaluation or score estimation.

Let $\mathcal{S}$ denote a continuous state space and let p_0 and p_1 be probability distributions over $\mathcal{S}$, representing a source (noise) distribution and a target (data) distribution, respectively. In the graph setting, $\mathcal{S}$ will later correspond to a continuous relaxation of adjacency matrices, node features, and edge features. Flow matching assumes access to paired samples (s_0, s_1) with $s_0 \sim p_0$ and $s_1 \sim p_1$, together with a family of interpolating states $(s_t)_{t\in[0,1]}$ defining a probability path between p_0 and p_1.

A commonly used choice is the *rectified linear path*,

$$s_t = (1-t)s_0 + ts_1, \qquad t \sim \mathcal{U}[0,1], \tag{10}$$

which yields a constant conditional velocity

$$u^\star(s_t \mid s_0, s_1) = s_1 - s_0. \tag{11}$$

Flow matching learns a time-conditioned vector field $v_\theta : \mathcal{S} \times [0,1] \to \mathcal{S}$, parameterised by θ, that approximates the conditional velocity along the path.

The FM objective is defined as a regression loss:

$$\mathcal{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{\substack{s_0\sim p_0,\, s_1\sim p_1 \\ t\sim\mathcal{U}[0,1]}} \left[\|v_\theta(s_t,t) - u^\star(s_t \mid s_0, s_1)\|_2^2 \right]. \tag{12}$$

Minimising Eq. (1) encourages the learned vector field to induce trajectories that transport mass from p_0 to p_1.

At generation time, new samples are obtained by solving the ordinary differential equation

$$\frac{ds_t}{dt} = v_\theta(s_t, t), \qquad s_{t=0} \sim p_0, \tag{13}$$

using a numerical integrator (e.g. Euler or Runge-Kutta), and taking $s_{t=1}$ as a sample from the learned model distribution.

C GNN Transformer for Learning Velocity Fields

At time $t \in [0,1]$, we represent the continuous graph state as $(A_t,\, X_t,\, F_t)$, where $A_t \in \mathbb{R}^{n\times n}$ is a soft adjacency matrix, $X_t \in \mathbb{R}^{n\times d_x}$ denotes node features, and $F_t \in \mathbb{R}^{n\times n\times d_f}$ denotes edge features.

The adjacency A_t is treated as a continuous variable during flow matching and serves as a structural bias rather than a discrete connectivity constraint.

Input projections and time conditioning. A scalar time $t \in [0, 1]$ is embedded via

$$c_t = \psi_t(t) \in \mathbb{R}^H, \tag{14}$$

where ψ_t is a Multi-Layer Perceptron (MLP). To enable joint node–edge reasoning, we map node and edge features into a common latent space of dimension H using separate learnable transformations. Specifically, node features are embedded via a function

$$\psi_x : \mathbb{R}^{d_x} \to \mathbb{R}^H, \qquad h_i^{(0)} = \psi_x\big(X_t(i)\big) + c_t,$$

while edge features are embedded via a distinct function

$$\psi_f : \mathbb{R}^{d_f+1} \to \mathbb{R}^H, \;\; b_{ij}^{(0)} = \psi_f\big([\, F_t(i,j) \parallel A_t(i,j) \,]\big) + c_t,$$

where $\parallel$ denotes concatenation. In practice, ψ_x and ψ_f are implemented as learnable linear layers or MLPs.

We denote the node and edge hidden states at layer ℓ by $h_i^{(\ell)} \in \mathbb{R}^H$ and $b_{ij}^{(\ell)} \in \mathbb{R}^H$. Given node features $X_t \in \mathbb{R}^{n \times d_x}$, edge features $F_t \in \mathbb{R}^{n \times n \times d_f}$, and adjacency $A_t \in \mathbb{R}^{n \times n}$, the GNN transformer predicts velocity fields (v_A, v_X, v_F) used for flow.

Stacked attention layers. We apply L layers of the edge-aware velocity attention:

$$h_i^{(\ell)}, b_{ij}^{(\ell)} = \mathrm{AttnLayer}\big(h_i^{(\ell-1)}, b_{ij}^{(\ell-1)}, A_t^{(\ell-1)}\big), \quad \ell = 1, \ldots, L. \tag{15}$$

Velocity prediction heads. Final velocity fields are obtained via linear projections:

$$v_X(i) = h_i^{(L)} W_{\text{out}}^X \in \mathbb{R}^{n \times d_x}, \tag{16}$$

$$v_F(i,j) = b_{ij}^{(L)} W_{\text{out}}^E \in \mathbb{R}^{d_f}, \tag{17}$$

$$v_A(i,j) = w_A^\top b_{ij}^{(L)} \in \mathbb{R}. \tag{18}$$

Symmetry enforcement. To ensure undirected graph consistency, we symmetrize outputs as

$$v_A \leftarrow \tfrac{1}{2}(v_A + v_A^\top), \qquad v_F(i,j) \leftarrow \tfrac{1}{2}\big(v_F(i,j) + v_F(j,i)\big). \tag{19}$$

This architecture jointly models node, edge, and adjacency dynamics under a unified time-conditioned attention mechanism. By explicitly propagating dense edge states and topology-aware attention biases, the model captures higher-order structural evolution, which is critical for flow matching on molecular graphs.

C.1 Edge-Aware Velocity Attention Layer

Structural-biased attention. We first compute query and key projections

$$Q_i^{(\ell)} = h_i^{(\ell)} W_Q^{(\ell)}, \qquad K_j^{(\ell)} = h_j^{(\ell)} W_K^{(\ell)}, \tag{20}$$

where $W_Q^{(\ell)}, W_K^{(\ell)} \in \mathbb{R}^{H \times H}$. The attention logits are defined as

$$S_{ij}^{(\ell)} = \frac{\langle Q_i^{(\ell)}, K_j^{(\ell)} \rangle}{\sqrt{H}} + A_t^{(\ell)}(i,j), \tag{21}$$

where the adjacency $A_t^{(\ell)}(i,j)$ acts as an additive structural bias. Row-wise normalized attention weights are then given by

$$\tilde{\alpha}_{ij}^{(\ell)} = \mathrm{softmax}_j(S_{ij}) = \frac{\exp(S_{ij}^{(\ell)})}{\sum_{k=1}^N \exp(S_{ik}^{(\ell)})}. \tag{22}$$

Edge-conditioned message construction. For each ordered node pair (i, j), we form a joint edge-conditioned concatenated representation

$$u_{ij}^{(\ell)} = [\, h_i^{(\ell)} \,\|\, h_j^{(\ell)} \,\|\, \psi(b_{ij}^{(\ell)}) \,] \in \mathbb{R}^{3H}, \tag{23}$$

where $\psi(\cdot)$ is a MLP, and then u_{ij} is processed by an edge MLP

$$m_{ij}^{(\ell)} = \phi_e(u_{ij}^{(\ell)}) \in \mathbb{R}^H. \tag{24}$$

Node aggregation and updates. Node-wise aggregated messages are computed as

$$\tilde{m}_i^{(\ell)} = \sum_{j=1}^{N} \tilde{\alpha}_{ij}^{(\ell)} m_{ij}^{(\ell)}, \tag{25}$$

and node states are updated via a MLP

$$h_i^{(\ell+1)} = h_i^{(\ell)} + \phi_n(\tilde{m}_i^{(\ell)}). \tag{26}$$

Edge states are updated symmetrically using

$$b_{ij}^{(\ell+1)} = b_{ij}^{(\ell)} + \phi_u(m_{ij}^{(\ell)}), \tag{27}$$

where ϕ_n and ϕ_u are learnable MLPs with SiLU activations.

Unlike standard attention layers that aggregate only node features, our formulation explicitly maintains and updates pairwise edge states. The adjacency bias $A_t^{(\ell)}(i, j)$ allows the model to softly favor plausible bonds while retaining the flexibility to modify connectivity during flow integration, which is essential for preserving cycles and long-range structural motifs in molecular graphs.

C.2 Distinction from Existing Graph Transformer Architectures

The GNN transformer above differs architecturally and semantically from existing graph transformers used in diffusion- and flow-based graph generation. DiGress [67] and DeFoG [55] operate on discrete noisy graphs and parameterise categorical denoising distributions or CTMC-based transition rates over node/edge states. VFM/CatFlow [19] uses relaxed categorical states but predicts a categorical endpoint distribution from which its vector field is induced. In contrast, our architecture makes four design choices absent from these models:

- **Joint continuous graph state and direct velocity prediction.** Flowette operates on the continuously relaxed graph state $G_t = (A_t, X_t, F_t)$ and directly predicts the coupled velocity field $v_\theta(G_t, t) = (v_A, v_X, v_F)$, where adjacency, node attributes, and edge attributes evolve simultaneously. These outputs are tangent directions that are repeatedly evaluated and integrated by the ODE solver, rather than categorical clean-state logits, endpoint probabilities, or discrete transition rates. The network must therefore remain meaningful throughout the continuum of fractional graph states encountered during generation, not only at discrete valid graphs.
- **Explicit separation of evolving topology and edge attributes.** Flowette treats soft connectivity $A_t(i, j)$ and edge attributes $F_t(i, j)$ as related but distinct evolving variables. The current soft adjacency is inserted directly into the attention logits, as in Eq. 21, so the evolving topology continuously modulates information exchange. In parallel, a separate pairwise hidden state $b_{ij}^{(\ell)}$, initialised from F_t, participates in message construction and is updated at every layer (Eqs. 23–27). Thus, A_t controls where and how strongly nodes communicate, while the pairwise state represents what edge-specific information is exchanged. This separation is particularly relevant for molecular graphs because bond existence and bond type are strongly related but are not identical variables.
- **Time conditions the complete topology–attribute dynamics.** The time embedding is injected into both node and pairwise representations, allowing the interaction between topology and attributes to change throughout transport. Near $t = 0$, the field performs coarse transformations from a graphette-prior sample, whereas near $t = 1$, it performs finer structural and attribute corrections. Time therefore does not merely serve as a denoising-step index; it conditions a single continuous vector field whose outputs (v_A, v_X, v_F) form the right-hand side of the graph ODE in Eq. 13.

- **Coupled and graph-consistent infinitesimal updates.** The final node states predict v_X, while the evolving pairwise states predict both v_A and v_F. Consequently, changes in atom representations, bond attributes, and connectivity are inferred from a shared evolving representation rather than from unrelated classification heads. The pairwise outputs are symmetrised for undirected graphs, and the complete vector field is permutation equivariant: $v_A(PAP^T, PX, PFP^T, t) = P\, v_A(A, X, F, t)\, P^T$, with the corresponding transformations for v_X and v_F. Therefore, node-relabelling consistency is respected at every infinitesimal ODE update, rather than being imposed only during final graph decoding.

Proposition C.1. *[Permutation equivariance of the velocity field] Let $v_\theta : (A_t, X_t, F_t, t) \mapsto \big(v_A,\ v_X,\ v_F\big)$ denote the time-conditioned GNN transformer described in Section 3, with L layers of edge-aware velocity attention, shared learnable linear maps and MLPs, and adjacency bias A_t inside the attention logits. For any permutation $\pi \in \mathfrak{S}_n$ with permutation matrix P_π, define the permuted graph state $(A'_t, X'_t, F'_t) = (P_\pi A_t P_\pi^\top,\ P_\pi X_t,\ P_\pi F_t P_\pi^\top)$. Then there exists a parameterization of v_θ such that $v_\theta(A'_t, X'_t, F'_t, t) = \big(P_\pi v_A P_\pi^\top,\ P_\pi v_X,\ P_\pi v_F P_\pi^\top\big)$, i.e., the predicted adjacency, node, and edge velocities transform equivariantly under any node relabeling.*

See Sec. E.2 for the proof.

D Chemistry-Aware Regularization

While the endpoint consistency loss $\mathcal{L}_{\text{end}}$ provides an endpoint-aware temporal reweighting of the velocity residual, it does not explicitly enforce chemical feasibility. We therefore introduce differentiable, chemistry-aware regularizers applied exclusively to the predicted endpoint $\hat{G}_1 = (\hat{A}_1, \hat{X}_1, \hat{F}_1)$, encouraging the model to produce globally valid molecular graphs at generation time.

Soft Valence Constraint. Let $A^w = \tilde{\sigma}\Big(\frac{1}{2}\Big(\hat{A}_1 + \hat{A}_1^\top\Big)\Big) \in \mathbb{R}_+^{n\times n}$ be a symmetric, non-negative weighted adjacency matrix, where $\tilde{\sigma}(\cdot) = \mathrm{clip}(\cdot, 0, 1)$ is applied entrywise. Let $\hat{F}_1(ij) \in \mathbb{R}^{\tilde{B}}$ be the bond-type logits for pair (i, j) and define bond-type probabilities

$$p_{ij} \;=\; \mathrm{softmax}(\hat{F}_1(ij)) \in \Delta^{\tilde{B}-1}, \tag{28}$$

where $\Delta^{\tilde{B}-1} = \{p \in \mathbb{R}^{\tilde{B}} \mid p \geq 0,\ \mathbf{1}^\top p = 1\}$. Let $b \in \mathbb{R}^{\tilde{B}}$ be the vector of bond orders (e.g., single/double/triple/aromatic), and define the expected bond order

$$\widehat{\mathrm{ord}}_{ij} \;=\; \langle p_{ij}, b\rangle \;=\; \sum_{k=1}^{\tilde{B}} p_{ij,k}\, b_k. \tag{29}$$

The expected valence of node i is then

$$\widehat{\mathrm{val}}_i \;=\; \sum_{j=1}^{n} A_{ij}^w\, \widehat{\mathrm{ord}}_{ij}. \tag{30}$$

Given the maximum allowable valence $\mathrm{val}_i^{\max}$ determined by the predicted atom type, we penalize soft violations via

$$\mathcal{L}_{\text{val}} = \frac{1}{n}\sum_{i=1}^{n} \max\big\{0, \widehat{\mathrm{val}}_i - \mathrm{val}_i^{\max}\big\}. \tag{31}$$

This loss couples adjacency and bond-type predictions in a fully differentiable manner, discouraging chemically implausible bonding patterns without imposing hard constraints.

Atom-type Marginal Matching. To prevent global drift in node-type composition (e.g., overproduction of common atoms), we align the predicted atom-type marginal with that of the target graph at the endpoint. Let d_a denote the number of atom-type categories, and let $X_{1,i,:d_a}$ denote the atom-type entries of the FGW-aligned target representation of node i. Define the predicted and target marginals

$$\hat{p} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{softmax}\big(\hat{X}_{1,i,:d_a}\big), \qquad \bar{p} = \frac{\bar{x}}{\mathbf{1}^\top \bar{x}}, \quad \bar{x} = \frac{1}{n}\sum_{i=1}^{n} X_{1,i,:d_a}, \tag{32}$$

and penalise their discrepancy as a per-entry MSE,

$$\mathcal{L}_{\text{atom}} = \frac{1}{d_a} \left\| \hat{p} - \bar{p} \right\|_2^2. \tag{33}$$

Unlike per-node classification losses, this term is permutation-invariant and operates purely at the graph level, enforcing global semantic consistency of the node-type composition. As a result, it complements the endpoint-aware velocity reweighting and improves overall chemical validity.

E Proofs for Structure Preserving Flows (Section 3)

E.1 Theoretical Properties of FGW-Coupling

Lemma E.1. *[Permutation invariance of FGW-distance] Let G_0, G_1 be two graphs as above, and let $\pi \in \mathfrak{S}_{n_0}$, $\tau \in \mathfrak{S}_{n_1}$ be arbitrary node permutations with permutation matrices P_π, P_τ. Define permuted graphs $\widetilde{G}_0 = (\pi \cdot A_0,\ \pi \cdot X_0,\ \pi \cdot F_0)$ and $\widetilde{G}_1 = (\tau \cdot A_1,\ \tau \cdot X_1,\ \tau \cdot F_1)$. Then, (1) The FGW-distance is invariant:* $\mathrm{FGW}_\alpha(G_0, G_1) = \mathrm{FGW}_\alpha(\widetilde{G}_0, \widetilde{G}_1)$*; (2) If $T^\star$ is an optimal coupling for* $\mathrm{FGW}_\alpha(G_0, G_1)$*, then $\widetilde{T}^\star := P_\pi T^\star P_\tau^\top$ is an optimal coupling for* $\mathrm{FGW}_\alpha(\widetilde{G}_0, \widetilde{G}_1)$*. Conversely, any optimal coupling $\widetilde{T}^\star$ for the permuted problem corresponds to an optimal coupling $T^\star = P_\pi^\top \widetilde{T}^\star P_\tau$ for the original problem.*

Proof. We first express how the costs transform under permutations.

For the appearance term, since π only reorders rows of X_0 and τ only reorders rows of X_1, we have

$$M(\pi \cdot X_0,\ \tau \cdot X_1) = P_\pi\, M(X_0, X_1)\, P_\tau^\top.$$

Similarly, the intra-graph distance matrices satisfy

$$C(\pi \cdot Z_0) = P_\pi\, C(Z_0)\, P_\pi^\top, \qquad C(\tau \cdot Z_1) = P_\tau\, C(Z_1)\, P_\tau^\top.$$

We now show that any feasible coupling T for the original problem corresponds to a feasible coupling $\widetilde{T}$ for the permuted problem with the same objective value.

Let $T \in \Pi(p, q)$, so $T\mathbf{1} = p$ and $T^\top \mathbf{1} = q$. Define $\widetilde{T} = P_\pi T P_\tau^\top$. Since P_π and P_τ are permutation matrices,

$$\widetilde{T}\mathbf{1} = P_\pi T P_\tau^\top \mathbf{1} = P_\pi T \mathbf{1} = P_\pi p = p,$$

because p is uniform and invariant under permutation. Analogously,

$$\widetilde{T}^\top \mathbf{1} = P_\tau T^\top P_\pi^\top \mathbf{1} = P_\tau T^\top \mathbf{1} = P_\tau q = q.$$

Non-negativity is preserved since P_π and P_τ just reorder entries. Thus $\widetilde{T} \in \Pi(p, q)$.

Feature cost. The linear appearance term for the permuted problem is

$$\begin{aligned}\langle \widetilde{T}, M(\pi \cdot X_0,\ \tau \cdot X_1) \rangle &= \langle P_\pi T P_\tau^\top,\ P_\pi M(X_0, X_1) P_\tau^\top \rangle \\ &= \langle T,\ M(X_0, X_1) \rangle,\end{aligned}$$

where we used invariance of the Frobenius inner product under simultaneous left- and right-multiplication by orthogonal matrices.

GW term. Let $C_0 = C(Z_0)$ and $C_1 = C(Z_1)$. For the permuted graphs, the structural costs are $\widetilde{C}_0 = P_\pi C_0 P_\pi^\top$ and $\widetilde{C}_1 = P_\tau C_1 P_\tau^\top$. The GW term of Eq. (3) for $(\widetilde{G}_0, \widetilde{G}_1)$ and coupling $\widetilde{T}$ is

$$\mathcal{L}_{\text{GW}}(\widetilde{T}; \widetilde{C}_0, \widetilde{C}_1) = \sum_{i,k} \sum_{j,\ell} \big(\widetilde{C}_0(i,k) - \widetilde{C}_1(j,\ell)\big)^2 \widetilde{T}_{ij} \widetilde{T}_{k\ell}.$$

Using the definitions of $\widetilde{C}_0, \widetilde{C}_1, \widetilde{T}$ and reindexing by $i' = \pi^{-1}(i)$, $k' = \pi^{-1}(k)$, $j' = \tau^{-1}(j)$, $\ell' = \tau^{-1}(\ell)$, we obtain

$$\mathcal{L}_{\text{GW}}(\widetilde{T}; \widetilde{C}_0, \widetilde{C}_1) = \mathcal{L}_{\text{GW}}(T; C_0, C_1).$$

Indeed, each term in the sum is just a relabeling of the indices of the corresponding term for $(T; C_0, C_1)$.

FGW Objective. Combining the two parts,

$$(1-\alpha)\,\langle \widetilde{T}, M(\pi \cdot X_0,\ \tau \cdot X_1)\rangle + \alpha\,\mathcal{L}_{\mathrm{GW}}(\widetilde{T}; \widetilde{C}_0, \widetilde{C}_1)$$

equals

$$(1-\alpha)\,\langle T, M(X_0, X_1)\rangle + \alpha\,\mathcal{L}_{\mathrm{GW}}(T; C_0, C_1).$$

Thus every feasible T for the original problem corresponds to a feasible $\widetilde{T}$ for the permuted problem with identical objective value, and conversely via $T = P_\pi^\top \widetilde{T} P_\tau$. Therefore the minimal values of the two optimization problems coincide, proving (1), and optimal solutions map to each other, proving (2). □

As a useful special case, if two graphs are identical up to a node relabeling, the FGW distance is zero and the optimal coupling is the corresponding permutation coupling.

Corollary E.2. *[Zero FGW-distance for isomorphic graphs] Assume $n_0 = n_1 = n$ and suppose there exists a permutation $\sigma \in \mathfrak{S}_n$ such that $x_i^0 = x_{\sigma(i)}^1$ and $C(Z_0)_{ik} = C(Z_1)_{\sigma(i),\sigma(k)}$ for all i, k. Let $T_\sigma = \frac{1}{n} P_\sigma$. Then T_σ is feasible, achieves objective value 0 in Eq.* (3)*, and hence $\mathrm{FGW}_\alpha(G_0, G_1) = 0$.*

Proof. By construction $T_\sigma \in \Pi(p, q)$ with uniform node measures. Under the stated assumptions, $M(X_0, X_1)$ and $C(Z_0), C(Z_1)$ align exactly under the permutation σ, so each term in both the appearance cost and the GW term of Eq. (3) vanishes when evaluated at T_σ. Non-negativity of the integrand implies that the minimal value of Eq. (3) is zero; hence T_σ is optimal and the distance is zero. □

Theorem E.3. *[Permutation-consistent FGW-coupling] Let $\{G_0^i\}_{i=1}^B$ and $\{G_1^j\}_{j=1}^B$ be two collections of graphs. Form the FGW-distance matrix $D \in \mathbb{R}_{\geq 0}^{B \times B}$ with $D_{ij} := \mathrm{FGW}_\alpha(G_0^i, G_1^j)$ as in Eq.* (4)*. Suppose there exists a permutation $\pi^\star \in \mathfrak{S}_B$ such that:*

- *For each i, the pair $(G_0^i, G_1^{\pi^\star(i)})$ satisfies the conditions of Corollary E.2, hence $D_{i,\pi^\star(i)} = 0$.*
- *For every i and every $j \neq \pi^\star(i)$ we have $D_{ij} \geq 0$, and the assignment $\pi^\star$ is the unique permutation achieving total cost zero: for any $\pi \neq \pi^\star$, $\sum_{i=1}^B D_{i,\pi(i)} > 0$.*

Then any solution of the Hungarian assignment problem

$$\widehat{\pi} \ \in\ \operatorname*{arg\,min}_{\pi \in \mathfrak{S}_B} \sum_{i=1}^{B} D_{i,\pi(i)} \tag{34}$$

recovers $\pi^\star$, i.e., $\widehat{\pi} = \pi^\star$. More generally, if there are several zero-cost permutations, the set of minimizers of Eq. (34) *coincides with the set of such zero-cost permutations.*

Proof. By construction of D, all entries are non-negative because both the appearance and GW terms in Eq. (3) are non-negative. Assumption (i) and Corollary E.2 imply that $D_{i,\pi^\star(i)} = 0$ for each i, hence

$$\sum_{i=1}^{B} D_{i,\pi^\star(i)} = 0.$$

Consider any other permutation $\pi \neq \pi^\star$. By assumption (ii), the corresponding total cost is strictly positive: $\sum_{i=1}^B D_{i,\pi(i)} > 0$. Therefore $\pi^\star$ is the unique minimizer of Eq. (34), and the Hungarian algorithm, which solves this linear assignment problem exactly, must return $\widehat{\pi} = \pi^\star$.

If multiple permutations achieve total cost zero, then any such permutation is a minimizer of Eq. (34), and no permutation with positive total cost can be optimal. Thus the set of minimizers is precisely the set of zero-cost permutations.

This theorem shows that, whenever the FGW-distance separates the correct pairings from all others, our Hungarian coupling step recovers a batch-wise one-to-one matching that is consistent with feature- and structure-preserving node permutations. □

E.2 Theoretical Analysis of the Velocity Field Predictor

Proposition C.1. *[Permutation equivariance of the velocity field] Let $v_\theta : (A_t, X_t, F_t, t) \mapsto \big(v_A,\ v_X,\ v_F\big)$ denote the time-conditioned GNN transformer described in Section 3, with L layers of edge-aware velocity attention, shared learnable linear maps and MLPs, and adjacency bias A_t inside the attention logits. For any permutation $\pi \in \mathfrak{S}_n$ with permutation matrix P_π, define the permuted graph state $(A'_t, X'_t, F'_t) = (P_\pi A_t P_\pi^\top,\ P_\pi X_t,\ P_\pi F_t P_\pi^\top)$. Then there exists a parameterization of v_θ such that $v_\theta(A'_t, X'_t, F'_t, t) = \big(P_\pi v_A P_\pi^\top,\ P_\pi v_X,\ P_\pi v_F P_\pi^\top\big)$, i.e., the predicted adjacency, node, and edge velocities transform equivariantly under any node relabeling.*

Proof. We prove the claim by induction on the L attention layers, using the fact that all linear maps have shared weights and that permutations act by row/column reordering.

Initialization. At time t, initial node and edge embeddings are

$$h_i^{(0)} = \psi_x(X_t(i)) + c_t, \qquad b_{ij}^{(0)} = \psi_f\big([F_t(i,j), A_t(i,j)]\big) + c_t,$$

where ψ_x and ψ_f are shared MLPs applied row-wise and c_t depends only on t. For the permuted state (A'_t, X'_t, F'_t) we obtain

$$h_i'^{(0)} = \psi_x(X'_t(i)) + c_t = \psi_x(X_t(\pi^{-1}(i))) + c_t = h^{(0)}_{\pi^{-1}(i)},$$

so in matrix form $H'^{(0)} = P_\pi H^{(0)}$. Similarly,

$$b_{ij}'^{(0)} = \psi_f([F'_t(i,j), A'_t(i,j)]) + c_t = \psi_f([F_t(\pi^{-1}(i), \pi^{-1}(j)), A_t(\pi^{-1}(i), \pi^{-1}(j))]) + c_t = b^{(0)}_{\pi^{-1}(i),\pi^{-1}(j)},$$

i.e., $B'^{(0)} = P_\pi B^{(0)} P_\pi^\top$.

Inductive step. Assume that at layer ℓ we have $H^{(\ell)} \in \mathbb{R}^{N\times H}$, $B^{(\ell)} \in \mathbb{R}^{N\times N\times H}$ and

$$H'^{(\ell)} = P_\pi H^{(\ell)}, \qquad B'^{(\ell)} = P_\pi B^{(\ell)} P_\pi^\top.$$

Attention logits. We compute

$$Q^{(\ell)} = H^{(\ell)} W_Q^{(\ell)}, \qquad K^{(\ell)} = H^{(\ell)} W_K^{(\ell)},$$

with shared weight matrices $W_Q^{(\ell)}, W_K^{(\ell)}$. For the permuted state,

$$Q'^{(\ell)} = H'^{(\ell)} W_Q^{(\ell)} = P_\pi H^{(\ell)} W_Q^{(\ell)} = P_\pi Q^{(\ell)},$$

and similarly $K'^{(\ell)} = P_\pi K^{(\ell)}$. The attention logits are

$$S_{ij}^{(\ell)} = \frac{\langle Q_i^{(\ell)}, K_j^{(\ell)} \rangle}{\sqrt{H}} + A_t(i,j),$$

and

$$S_{ij}'^{(\ell)} = \frac{\langle Q_i'^{(\ell)}, K_j'^{(\ell)} \rangle}{\sqrt{H}} + A'_t(i,j).$$

Using $Q'^{(\ell)} = P_\pi Q^{(\ell)}$, $K'^{(\ell)} = P_\pi K^{(\ell)}$, and $A'_t = P_\pi A_t P_\pi^\top$, we obtain

$$S'^{(\ell)} = P_\pi S^{(\ell)} P_\pi^\top.$$

Row-wise softmax is equivariant under row permutations, so the attention weights satisfy

$$\alpha'^{(\ell)} = P_\pi \alpha^{(\ell)} P_\pi^\top.$$

Edge messages. For each pair (i,j), we compute

$$u_{ij}^{(\ell)} = \big[h_i^{(\ell)} \,\|\, h_j^{(\ell)} \,\|\, \psi(b_{ij}^{(\ell)})\big], \qquad m_{ij}^{(\ell)} = \phi_e(u_{ij}^{(\ell)}),$$

where ψ and ϕ_e are shared MLPs. By the inductive hypothesis and shared weights,

$$u_{ij}'^{(\ell)} = u^{(\ell)}_{\pi^{-1}(i),\pi^{-1}(j)}, \quad m_{ij}'^{(\ell)} = m^{(\ell)}_{\pi^{-1}(i),\pi^{-1}(j)}.$$

Node aggregation. Node-wise aggregated messages are

$$\widetilde{m}_i^{(\ell)} = \sum_{j=1}^{N} \alpha_{ij}^{(\ell)} m_{ij}^{(\ell)}.$$

Using the permutation relations for $\alpha^{(\ell)}$ and $m^{(\ell)}$, one checks that $\widetilde{m}_i'^{(\ell)} = \widetilde{m}_{\pi^{-1}(i)}^{(\ell)}$, i.e., $\widetilde{M}'^{(\ell)} = P_\pi \widetilde{M}^{(\ell)}$.

Node and edge updates. Node states are updated via a shared MLP ϕ_n:

$$h_i^{(\ell+1)} = h_i^{(\ell)} + \phi_n(\widetilde{m}_i^{(\ell)}).$$

Thus

$$h_i'^{(\ell+1)} = h_{\pi^{-1}(i)}^{(\ell)} + \phi_n(\widetilde{m}_{\pi^{-1}(i)}^{(\ell)}) = h_{\pi^{-1}(i)}^{(\ell+1)},$$

so $H'^{(\ell+1)} = P_\pi H^{(\ell+1)}$. Edge states are updated symmetrically via a shared MLP ϕ_u:

$$b_{ij}^{(\ell+1)} = b_{ij}^{(\ell)} + \phi_u(m_{ij}^{(\ell)}),$$

which yields $B'^{(\ell+1)} = P_\pi B^{(\ell+1)} P_\pi^\top$.

By induction, the equivariance relations hold for all $\ell = 0, \ldots, L$.

Output velocities. Finally, the velocity heads are linear maps applied row-wise:

$$v_X(i) = h_i^{(L)} W_{\text{out}}^X, \quad v_F(i,j) = b_{ij}^{(L)} W_{\text{out}}^E, \quad v_A(i,j) = w_A^\top b_{ij}^{(L)}.$$

Therefore, for the permuted state, we have

$$v'_X(i) = v_X(\pi^{-1}(i)), \quad v'_F(i,j) = v_F(\pi^{-1}(i), \pi^{-1}(j)), \quad v'_A(i,j) = v_A(\pi^{-1}(i), \pi^{-1}(j)),$$

or in matrix form

$$v'_X = P_\pi v_X, \qquad v'_F = P_\pi v_F P_\pi^\top, \qquad v'_A = P_\pi v_A P_\pi^\top.$$

This is precisely the desired permutation equivariance. □

Together, Lemma E.1 and Theorem E.3 and Proposition C.1 show that our FGW coupling, Hungarian pairing, and velocity field respect node permutations, ensuring that the overall flow-matching objective is well-defined on graphs up to isomorphism.

E.3 Theoretical Properties of the Learning Objective

E.3.1 Derivation: $\mathcal{L}_{\text{end}}$ as a Reweighted Velocity Residual

We show explicitly that $\mathcal{L}_{\text{end}}$ (Eq. (9)) is not an independent supervision signal but an algebraically induced, non-uniform temporal reweighting of the velocity residual already penalised by $\mathcal{L}_{\text{vel}}$.

Recall the rectified interpolant $A_t = (1-t)A_0 + tA_1$ and the ideal constant velocity target $\Delta A = A_1 - A_0$. Substituting the interpolant into the one-step endpoint prediction (Eq. (8)):

$$\begin{aligned} \widehat{A}_1 &= A_t + (1-t)v_A \\ &= \big[(1-t)A_0 + tA_1\big] + (1-t)v_A. \end{aligned} \tag{35}$$

Subtracting the true target A_1:

$$\begin{aligned} \widehat{A}_1 - A_1 &= (1-t)A_0 + tA_1 - A_1 + (1-t)v_A \\ &= (1-t)A_0 - (1-t)A_1 + (1-t)v_A \\ &= (1-t)\big(A_0 - A_1\big) + (1-t)v_A \\ &= (1-t)\big(v_A - (A_1 - A_0)\big) \\ &= (1-t)(v_A - \Delta A). \end{aligned} \tag{36}$$

The same relation holds identically for X and F. Squaring Eq. (36) gives

$$\|\widehat{A}_1 - A_1\|_F^2 = (1-t)^2 \|v_A - \Delta A\|_F^2, \tag{37}$$

Since the same per-entry normalisations and component weights are used in Eqs. (7) and (9), the relation holds for the complete velocity and endpoint losses:

$$\mathcal{L}_{\mathrm{end}}(t) = (1-t)^2 \mathcal{L}_{\mathrm{vel}}(t). \tag{38}$$

Consequently, their combined contribution to the training objective is

$$\mathcal{L}_{\mathrm{vel}}(t) + \beta_{\mathrm{end}} \mathcal{L}_{\mathrm{end}}(t) = \left[1 + \beta_{\mathrm{end}}(1-t)^2\right] \mathcal{L}_{\mathrm{vel}}(t). \tag{39}$$

Thus, $\mathcal{L}_{\mathrm{end}}$ introduces no additional transport target; it acts as an explicit time-dependent reweighting of the same velocity residual.

The weight $(1-t)^2$ has a direct interpretation: $(1-t)$ is the remaining extrapolation horizon from time t to $t=1$ in the one-step endpoint estimate. Consequently, the same velocity residual induces a larger predicted endpoint error when evaluated earlier in the trajectory (t close to 0) than later (t close to 1). Combining $\mathcal{L}_{\mathrm{vel}}$ with $\beta_{\mathrm{end}}\mathcal{L}_{\mathrm{end}}$ therefore provides an endpoint-aware temporal reweighting that places greater emphasis on velocity residuals with a larger effect on the predicted terminal state, without introducing any new transport target beyond ΔA, ΔX, and ΔF. Under the assumptions of Theorem E.4, driving the velocity residual to zero also drives both $\mathcal{L}_{\mathrm{vel}}$ and $\mathcal{L}_{\mathrm{end}}$ to zero, since the two losses differ only by the multiplicative factor $(1-t)^2$.

Theorem E.4. *[Endpoint Consistency of Rectified Transport] Fix an FGW-coupled pair of graphs (A_0, X_0, F_0) and (A_1, X_1, F_1), and let $(A_t, X_t, F_t)_{t\in[0,1]}$ denote the corresponding rectified path. Assume $\lambda_x, \lambda_e > 0$ and that $t \mapsto v_\theta(A_t, X_t, F_t, t)$ is continuous along this path. If $\mathbb{E}_{t\sim\mathcal{U}[0,1]}\left[\mathcal{L}_{\mathrm{vel}}(t)\right] = 0$, then*

$$v_\theta(A_t, X_t, F_t, t) = (\Delta A, \Delta X, \Delta F), \qquad \forall\, t \in [0,1]. \tag{40}$$

Consequently, the rectified path satisfies

$$\frac{d}{dt}(A_t, X_t, F_t) = v_\theta(A_t, X_t, F_t, t), \qquad (A_0, X_0, F_0) \text{ given}, \tag{41}$$

and reaches the target endpoint

$$(A_1, X_1, F_1) = (A_0 + \Delta A,\ X_0 + \Delta X,\ F_0 + \Delta F). \tag{42}$$

Moreover, for any $K \in \mathbb{N}$, K-step explicit Euler integration with step size $h = 1/K$, initialized at (A_0, X_0, F_0), recovers the exact endpoint (A_1, X_1, F_1).

Proof. We prove the result in two steps.

Step 1: Identification of the velocity field. Let

$$s_t = (A_t, X_t, F_t), \qquad \Delta = (\Delta A, \Delta X, \Delta F),$$

where, by definition of the rectified path,

$$A_t = (1-t)A_0 + tA_1, \qquad X_t = (1-t)X_0 + tX_1, \qquad F_t = (1-t)F_0 + tF_1.$$

Hence

$$\Delta A = A_1 - A_0, \qquad \Delta X = X_1 - X_0, \qquad \Delta F = F_1 - F_0.$$

By Eq. (7), the velocity-matching loss at time t is

$$\begin{aligned}\mathcal{L}_{\mathrm{vel}}(t) = \frac{1}{N_A}\left\|v_A(A_t, X_t, F_t, t) - \Delta A\right\|_F^2 + \lambda_x \frac{1}{N_X}\left\|v_X(A_t, X_t, F_t, t) - \Delta X\right\|_F^2 \\ + \lambda_e \frac{1}{N_F}\left\|v_F(A_t, X_t, F_t, t) - \Delta F\right\|_F^2.\end{aligned} \tag{43}$$

Since $N_A, N_X, N_F > 0$ and, by assumption, $\lambda_x, \lambda_e > 0$, every term in Eq. (43) is nonnegative and has a strictly positive coefficient.

Moreover, by assumption, $t \mapsto v_\theta(A_t, X_t, F_t, t)$ is continuous along the rectified path. Since the squared Frobenius norm is continuous, $\mathcal{L}_{\text{vel}}(t)$ is therefore a continuous, nonnegative function on $[0, 1]$.

For $t \sim \mathcal{U}[0, 1]$, the hypothesis

$$\mathbb{E}_{t\sim\mathcal{U}[0,1]}\big[\mathcal{L}_{\text{vel}}(t)\big] = 0$$

is equivalent to

$$\int_0^1 \mathcal{L}_{\text{vel}}(t)\, dt = 0.$$

A continuous nonnegative function on $[0, 1]$ can have zero integral only if it vanishes everywhere. Hence

$$\mathcal{L}_{\text{vel}}(t) = 0, \qquad \forall\, t \in [0, 1]. \tag{44}$$

Because all coefficients in Eq. (43) are strictly positive, Eq. (44) implies that each squared residual vanishes individually. Therefore, for every $t \in [0, 1]$,

$$v_A(A_t, X_t, F_t, t) = \Delta A, \qquad v_X(A_t, X_t, F_t, t) = \Delta X, \qquad v_F(A_t, X_t, F_t, t) = \Delta F. \tag{45}$$

Thus, along the entire rectified path, the learned velocity field coincides with the ideal constant transport $(\Delta A, \Delta X, \Delta F)$.

Step 2: Exact reconstruction by the ODE and Euler integration. By construction of the rectified path,

$$\frac{dA_t}{dt} = \Delta A, \qquad \frac{dX_t}{dt} = \Delta X, \qquad \frac{dF_t}{dt} = \Delta F. \tag{46}$$

Combining these identities with Eq. (45) gives

$$\frac{d}{dt}(A_t, X_t, F_t) = v_\theta(A_t, X_t, F_t, t)$$

for every $t \in [0, 1]$. Hence the rectified path satisfies the learned ODE with initial condition (A_0, X_0, F_0).

Integrating from $t = 0$ to $t = 1$ yields

$$A_1 = A_0 + \int_0^1 \Delta A\, dt = A_0 + \Delta A, \tag{47}$$

$$X_1 = X_0 + \int_0^1 \Delta X\, dt = X_0 + \Delta X, \tag{48}$$

$$F_1 = F_0 + \int_0^1 \Delta F\, dt = F_0 + \Delta F. \tag{49}$$

Thus the rectified ODE trajectory reaches exactly the target endpoint (A_1, X_1, F_1).

It remains to show that explicit Euler integration is also exact. Let

$$t_k = \frac{k}{K}, \qquad h = \frac{1}{K}, \qquad k = 0, \ldots, K,$$

and initialize

$$(A^{(0)}, X^{(0)}, F^{(0)}) = (A_0, X_0, F_0).$$

The Euler updates are

$$A^{(k+1)} = A^{(k)} + h\, v_A(A^{(k)}, X^{(k)}, F^{(k)}, t_k), \tag{50}$$

$$X^{(k+1)} = X^{(k)} + h\, v_X(A^{(k)}, X^{(k)}, F^{(k)}, t_k), \tag{51}$$

$$F^{(k+1)} = F^{(k)} + h\, v_F(A^{(k)}, X^{(k)}, F^{(k)}, t_k). \tag{52}$$

We show by induction that the Euler iterate coincides with the rectified state at every grid point:

$$(A^{(k)}, X^{(k)}, F^{(k)}) = (A_{t_k}, X_{t_k}, F_{t_k}), \qquad k = 0, \ldots, K. \tag{53}$$

For $k = 0$, Eq. (53) holds by initialization. Suppose it holds for some $k < K$. Then

$$(A^{(k)}, X^{(k)}, F^{(k)}) = (A_{t_k}, X_{t_k}, F_{t_k}),$$

and therefore, by Eq. (45),

$$v_\theta(A^{(k)}, X^{(k)}, F^{(k)}, t_k) = (\Delta A, \Delta X, \Delta F).$$

The Euler update consequently gives

$$A^{(k+1)} = A_{t_k} + h\Delta A = A_0 + (t_k + h)\Delta A = A_{t_{k+1}}, \tag{54}$$

$$X^{(k+1)} = X_{t_k} + h\Delta X = X_0 + (t_k + h)\Delta X = X_{t_{k+1}}, \tag{55}$$

$$F^{(k+1)} = F_{t_k} + h\Delta F = F_0 + (t_k + h)\Delta F = F_{t_{k+1}}, \tag{56}$$

where $t_{k+1} = t_k + h$. Hence Eq. (53) holds at step $k + 1$, completing the induction.

Taking $k = K$ gives $t_K = 1$, and therefore

$$(A^{(K)}, X^{(K)}, F^{(K)}) = (A_{t_K}, X_{t_K}, F_{t_K}) = (A_1, X_1, F_1).$$

Thus K-step explicit Euler integration with $h = 1/K$ recovers the exact endpoint for every $K \in \mathbb{N}$. □

By Eq. (38), the conclusion also implies $\mathcal{L}_{\text{end}}(t) = 0$ for every $t \in [0, 1]$; thus the endpoint term introduces no additional zero-loss condition beyond $\mathcal{L}_{\text{vel}}$.

Theorem E.5. *[Finite-Step Stability] Let $s_t = (A_t, X_t, F_t)$ denote the rectified path between s_0 and s_1, and let $v^\star(s_t, t) = (\Delta A, \Delta X, \Delta F)$ be the ideal constant velocity. Suppose v_θ satisfies*

$$\|v_\theta(s_t, t) - v^\star(s_t, t)\|_F \le \varepsilon \quad \textit{for all } t \in [0, 1], \tag{57}$$

and is L-Lipschitz in s, i.e., $\|v_\theta(s, t) - v_\theta(s', t)\|_F \le L\|s - s'\|_F, \forall s, s',\ \forall t \in [0, 1]$. Let $s^{(K)}$ denote the endpoint obtained by K-step explicit Euler integration of v_θ from s_0 with step size $h = 1/K$. Then there exists a constant $C > 0$, depending only on L, such that

$$\|s^{(K)} - s_1\|_F \le C\varepsilon. \tag{58}$$

Proof. Let $s_t^\star$ denote the exact rectified path driven by $v^\star$, i.e., $s_t^\star = s_0 + t(s_1 - s_0)$, so that $s_1^\star = s_1$ and $ds_t^\star/dt = v^\star(s_t^\star, t)$. Let $s^{(k)}$ be the Euler iterates with step size $h = 1/K$, starting at $s^{(0)} = s_0$:

$$s^{(k+1)} = s^{(k)} + h\, v_\theta(s^{(k)}, t_k), \qquad t_k = kh. \tag{59}$$

Define the error at step k as

$$e_k := s^{(k)} - s^\star_{t_k}. \tag{60}$$

We derive a recursion for e_k. Using the definition of $s^{(k+1)}$ and the exact flow of $s_t^\star$,

$$e_{k+1} = s^{(k+1)} - s^\star_{t_{k+1}} \tag{61}$$

$$= s^{(k)} + h\, v_\theta(s^{(k)}, t_k) - \left(s^\star_{t_k} + \int_{t_k}^{t_{k+1}} v^\star(s_u^\star, u)\, du \right) \tag{62}$$

$$= e_k + h\big(v_\theta(s^{(k)}, t_k) - v^\star(s^\star_{t_k}, t_k)\big) - \int_{t_k}^{t_{k+1}} \big(v^\star(s_u^\star, u) - v^\star(s^\star_{t_k}, t_k)\big)\, du. \tag{63}$$

Taking Frobenius norms and applying the triangle inequality,

$$\|e_{k+1}\|_F \le \|e_k\|_F + h\, \|v_\theta(s^{(k)}, t_k) - v^\star(s^\star_{t_k}, t_k)\|_F + \int_{t_k}^{t_{k+1}} \|v^\star(s_u^\star, u) - v^\star(s^\star_{t_k}, t_k)\|_F\, du. \tag{64}$$

Velocity mismatch term. Add and subtract $v_\theta(s^\star_{t_k}, t_k)$, and use the Lipschitz property and Eq. (57):

$$\begin{aligned}
&\|v_\theta(s^{(k)}, t_k) - v^\star(s^\star_{t_k}, t_k)\|_F \\
&\quad \le \|v_\theta(s^{(k)}, t_k) - v_\theta(s^\star_{t_k}, t_k)\|_F + \|v_\theta(s^\star_{t_k}, t_k) - v^\star(s^\star_{t_k}, t_k)\|_F \qquad (65)\\
&\quad \le L\|s^{(k)} - s^\star_{t_k}\|_F + \varepsilon = L\|e_k\|_F + \varepsilon. \qquad (66)
\end{aligned}$$

Local truncation term. Since $v^\star$ is constant along the rectified path (indeed, $v^\star(s^\star_u, u) = \Delta s := s_1 - s_0$ for all u), we have

$$v^\star(s^\star_u, u) - v^\star(s^\star_{t_k}, t_k) = 0 \quad \text{for all } u \in [t_k, t_{k+1}], \qquad (67)$$

and therefore the integral term in Eq. (64) vanishes exactly.

Putting it together. Substituting the above bounds into Eq. (64) yields

$$\|e_{k+1}\|_F \le \|e_k\|_F + h\big(L\|e_k\|_F + \varepsilon\big) = (1 + Lh)\|e_k\|_F + h\varepsilon. \qquad (68)$$

Since $e_0 = s^{(0)} - s^\star_0 = 0$, for $L > 0$ we obtain

$$\begin{aligned}
\|e_K\|_F &\le h\varepsilon \sum_{j=0}^{K-1} (1 + Lh)^j \qquad (69)\\
&= h\varepsilon \frac{(1 + Lh)^K - 1}{Lh} \qquad (70)\\
&= \frac{\varepsilon}{L}\big((1 + Lh)^K - 1\big). \qquad (71)
\end{aligned}$$

Since $K = 1/h$ and $(1 + Lh)^{1/h} \le e^L$ for $h \in (0, 1]$,

$$(1 + Lh)^K = (1 + Lh)^{1/h} \le e^L, \qquad (72)$$

and hence

$$\|e_K\|_F \le \frac{e^L - 1}{L}\,\varepsilon. \qquad (73)$$

For $L = 0$, the recursion reduces to

$$\|e_{k+1}\|_F \le \|e_k\|_F + h\varepsilon,$$

which gives

$$\|e_K\|_F \le Kh\varepsilon = \varepsilon.$$

Thus, with

$$C = \begin{cases} \dfrac{e^L - 1}{L}, & L > 0, \\ 1, & L = 0, \end{cases}$$

we obtain

$$\|s^{(K)} - s_1\|_F = \|e_K\|_F \le C\varepsilon, \qquad (74)$$

as claimed. □

Theorem E.5 formalizes how a uniform velocity approximation error controls the endpoint error under finite-step integration. We now relate this raw Frobenius error to the per-entry MSE used in our training objective. For $\lambda_x, \lambda_e > 0$, let

$$N_{\max} = \max\left\{N_A,\ \frac{N_X}{\lambda_x},\ \frac{N_F}{\lambda_e}\right\}. \qquad (75)$$

From Eq. (7),

$$\|v_\theta(s_t, t) - v^\star(s_t, t)\|_F^2 \le N_{\max}\,\mathcal{L}_{\text{vel}}(t). \qquad (76)$$

Hence, if $\mathcal{L}_{\text{vel}}(t) \le \delta$ for all $t \in [0, 1]$, then Eq. (57) holds with

$$\varepsilon = \sqrt{N_{\max}\,\delta}, \qquad (77)$$

and Theorem E.5 gives $\|s^{(K)} - s_1\|_F \le C\sqrt{N_{\max}\delta}$. As shown in Sec. E.3.1, $\mathcal{L}_{\text{end}}$ provides an endpoint-aware $(1-t)^2$ temporal reweighting of the same velocity residual rather than an independent error signal.

F Computational Complexity Analysis

Let B be the batch size, n the graph size used for flow matching, m the number of edges $|E|$ in the graph, L the number of transformer layers, H the hidden dimension, and K the number of Euler steps.

FGW coupling. Computing the FGW cost matrix between B source and B target graphs in a single minibatch requires B^2 FGW solves. Using a square-loss FGW solver with I iterations, this costs $O(B^2 I n^3)$, plus $O(B^2 m(d_x + d_z))$ to form feature and structure costs; Hungarian matching adds $O(B^3)$. Under the default *online* protocol (Sec. 3.1), this cost is paid at every minibatch of every epoch. Under the *fixed-pool* protocol, it is instead a one-time preprocessing cost, computed once in $B \times B$ blocks over the training set and reused across all subsequent epochs; *periodic* coupling amortises this cost by recomputing it only every $\tilde{\mathcal{R}}$ epochs. All three variants are embarrassingly parallel across blocks. Sec. J.8 reports empirical runtimes for all three protocols.

Flow-matching training. After FGW precomputation, each minibatch contains B matched graph pairs. Edge-aware flow matching costs $O(LmH^2 + md_f)$ per pair, yielding a per-minibatch cost $O(B(LmH^2 + md_f))$. Training over $\mathcal{R}$ epochs and M minibatches therefore costs $O(\mathcal{R}M\, B(LmH^2 + md_f))$.

Sampling. Euler integration for K steps requires K forward passes of the transformer, with total cost $O(Km(LH^2 + d_f))$.

G Generative stories

G.1 Generative story for graphons, sparsified graphons and graphettes

Algorithm 2 Sampling from a graphon W

input n, W
output Adjacency matrix A of sampled graph.

Sample $x_1, \ldots, x_n \overset{iid}{\sim} \text{Unif}[0,1]$
for $i \in \{1, \ldots, n\}$ **do**
  for $j \in \{(i+1), \ldots, n\}$ **do**
    Sample $A_{ij} \sim \text{Bernoulli}(W(x_i, x_j))$
    Set $A_{ji} = A_{ij}$
  end for
end for
return A

Algorithm 3 Sampling from a sparsified graphon $W_n = \rho_n W$

input n, W, sequence $\{\rho_i\}_i = (\rho_1, \rho_2, \ldots, \rho_n, \ldots)$
output Adjacency matrix A of sampled graph.

Get the nth value of $\{\rho_i\}_i$ denoted by ρ_n
Sample $x_1, \ldots, x_n \overset{iid}{\sim} \text{Unif}[0,1]$
for $i \in \{1, \ldots, n\}$ **do**
  for $j \in \{(i+1), \ldots, n\}$ **do**
    Sample $A_{ij} \sim \text{Bernoulli}(\rho_n W(x_i, x_j))$
    Set $A_{ji} = A_{ij}$
  end for
end for
return A

Algorithm 4 Sampling from a graphette $\mathcal{W} = (W, \rho_n, f)$

input n, W, sequence $\{\rho_i\}_i = (\rho_1, \rho_2, \ldots, \rho_n, \ldots)$, graph edit function $f(G, \cdots)$ where $\cdots$ denote other function specific parameters
output Adjacency matrix A of sampled graph.

// *First sample a graph from the sparsified graphon*
Get the nth value of $\{\rho_i\}_i$ denoted by ρ_n
Sample $x_1, \ldots, x_n \overset{iid}{\sim} \text{Unif}[0,1]$
for $i \in \{1, \ldots, n\}$ **do**
  for $j \in \{(i+1), \ldots, n\}$ **do**
    Sample $A'_{ij} \sim \text{Bernoulli}(\rho_n W(x_i, x_j))$
    Set $A'_{ji} = A'_{ij}$
  end for
end for
Get the graph G' from the adjacency matrix A';
// *Then enact the graph edit function on it*
$G = f(G', \cdots)$;
Get the adjacency matrix A of G;
return A

H Graph edit functions and theorem proofs in Section 4

H.1 Graph edit functions GEF 1 - GEF 4

We use the following four graph edit functions in our experiments.

GEF 1: ***Identity***. *Let $f = I$ be the identity function with $I(G) = G$, which does not edit the graph.*

The identity graph edit function is used in Lemmas H.2 and H.3 to show that graphettes can recover graphons and sparsified graphons. Furthermore, for stochastic block models, as illustrated in Section 5.2, graphettes are equivalent to graphons and our Flowette model achieves competitive performance, including state-of-the-art results on several metrics across multiple benchmarks.

GEF 2: ***Cycle deletion***. *Let $f = h$ denote a global function that removes cycles from the graph.*

When the application requires that graphs are trees (no cycles), we choose $f = h$. For a connected graph the resulting graph is a tree.

The class of graphettes also includes graph edit functions which may be parametrised, for example we may wish to model different ring sizes in molecular graphs in Sec. 5.3.

GEF 3: ***Ring addition***. *Let $R(p, c)$ denote a function where a ring of size c is added and connected to a node i with probability p, i.e., $R(p, c) = \text{Bernoulli}(p) \times c$.*

For social network graphs, we model hubs, which are highly connected individuals.

GEF 4: ***Star addition***. *Let $S\left(a, b, \{u_i\}_{i=1}^n\right)$ denote a function that adds stars given by a Poisson random variable $\text{Poi}(an\exp(u_i + b))$ to a node i where $\{u_i\}_{i=1}^n = (u_1, \ldots, u_n)$, $0 \leq u_i \leq 1$, $a > 0$ and $|b| < 1$.*

This models a non-uniform probability of any particular node i (indexed by u_i) , having a star added, with star sizes varying with u_i.

H.2 Theorem proofs in Section 4

Lemma H.1. *The graphette $\mathcal{W} = (W, \rho_n, f)$ recovers W when $\rho_n = 1$ and f is the identity function.*

Proof. This results in G being sampled from $W_n = W$ and as f is the identity, no edits are done. For increasing n the resulting graph sequence $\{G_n\}_n$ is dense. □

Lemma H.2. *The graphette $\mathcal{W} = (W, \rho_n, f)$ generates dense graphs from a family of graphons $\{W_n\}_n$ when $\rho_n \to c > 0$ and f is the identity function.*

Proof. When $\rho_n \to c > 0$, the graphon family $\{W_n\}_n$ where $W_n = \rho_n W \not\to 0$. As graphs sampled from any non-zero graphon are dense we have the result. □

Lemma H.3. *The graphette $\mathcal{W} = (W, \rho_n, f)$ recovers the sparsified graphon W_n when $\rho_n \to 0$ and f is the identity function.*

Proof. When $\rho_n \to 0$ and f is the identity, $\mathcal{W}$ generates graphs from $W_n = \rho_n W$ and $W_n \to 0$ as in the sparsified graphon setting. □

Lemma H.4. *The graphette $\mathcal{W} = (W, \rho_n, f)$ can generate graphs with stars and isolated edges similar to the graphex construction.*

Proof. Let $\rho_n = 1$. As illustrated in [66], a star function such as $S(x, n) = \text{Poi}(n\exp(x + 0.2))$, where $S(x, n)$ edges are added to the node at latent position x can generate stars. An isolated edge function such as $I(\ell, n) = \text{Poi}(n\ell)$ can add isolated edges at a rate ℓ to the graph. By letting $f = I \circ S$, both stars and isolated edges can be added. □

Lemma H.5. *The graphette $\mathcal{W} = (W, \rho_n, f)$ with $\rho_n = 1$ and an appropriate star function can produce sparse graphs.*

Proof. Consider a simple deterministic star function $S(x, n) = \lfloor \alpha n \rfloor$, where $\alpha > 0$. As $\rho_n = 1$, the graph is first sampled from W. Note $\{G'_n\}_n$ sampled from W is dense. The star function will add $\sum_{i=1}^{n} \lfloor \alpha n \rfloor \approx \alpha n^2$ new nodes and edges to G_n. After star addition, the edge density of G_n

$$\text{density}(G_n) \approx \frac{m + \alpha n^2}{(n + \alpha n^2)^2} \to 0\,,$$

making the sequence sparse, where m denotes the edges in G'_n before enacting the star function. For stochastic star functions such as $S(x, n) = \text{Poi}(\alpha n)$ or $S(x, n) = \text{Poi}(\alpha n \exp(bx - c))$ the result holds with high probability for $\alpha, b > 0$ and can be shown using concentration inequalities. □

Lemma H.6. *The graphette $\mathcal{W} = (W, \rho_n, f)$ with $\rho_n = 1$ and ring function $R(p, c)$ (GEF 3) produces dense graphs with high probability.*

Proof. The graph sequence $\{G'_n\}_n$ is sampled from $W_n = W$ and is dense. Suppose G'_n has n nodes and m edges. Then the expectation of the number of nodes to be replaced by rings is np. These nodes will be replaced by npc nodes resulting in an increase in node count from n to $n + np(c - 1)$ in expectation. As a ring of size c has c edges, the edge count increases from m to $m + npc$ in expectation. Using the first order Taylor approximation of edge density we obtain

$$\mathbb{E}\left(\text{density}(G_n)\right) \approx \frac{m + npc}{\frac{1}{2}(n + np(c-1))(n + np(c-1) - 1)}\,.$$

As graphs sampled from W are dense, $2m/n(n-1) \to \rho \neq 0$. This implies that $\mathbb{E}\left(\text{density}(G_n)\right)$ converges to a non-zero constant. In expectation the graphs $\{G_n\}_n$ are dense. Using concentration bounds we can show that the probability of the number of nodes and edges in G_n being away from their expectations is small. Thus, we can show that $P(\text{density}(G_n) < \mathbb{E}\left(\text{density}(G_n)\right) - \epsilon)$ is small giving us the result. □

Lemma H.7. *The graphette $\mathcal{W} = (W, \rho_n, f)$ with $\rho_n \to 0$ and ring function $R(x_i, p, c)$ (graph edit function 4.1.3) with $\mathcal{C} = \{c\}$ produces sparse graphs for $c \geq 3$.*

Proof. Consider a connected G'_n. Suppose G'_n has n nodes and m edges and suppose a graph edit replaces k nodes in a graph G'_n with rings of size c resulting in graph G_n. Then G_n, the edited graph has $n - k + kc$ nodes and $m + kc$ edges. Thus, the new edge density is

$$\text{density}(G_n) = \frac{m + kc}{\frac{1}{2}(n + k(c-1))(n + k(c-1) - 1)}\,.$$

By computing $\text{density}(G'_n) - \text{density}(G_n)$ we can show that

$$\text{density}(G_n) \leq \text{density}(G'_n)$$

for $c \geq 3$ showing that the edge density decreases by the addition of rings of size c. As graphs sampled from $W_n = \rho_n W$ are sparse we obtain the result. □

Theorem 4.5. *Let $\mathcal{W} = (W, \rho_n, f)$ denote a graphette and let $G' \sim \mathsf{G}(n, W, \rho_n)$ and $G \sim f(G')$ where $f \in \{R, S\}$ (GEF 3, 4) with ring size $c > 3$ for R. Suppose $|V(G')| = n$ and $|V(G)| = n + m$. Then for any triangle-covered $F \in \mathcal{F}$, $\hom(F, G) = \hom(F, G')$ and $t(F, G) = \frac{n^{|V(F)|}}{(n+m)^{|V(F)|}} t(F, G')$.*

Proof. As $G' \subseteq G$ we have $\text{Hom}(F, G') \subseteq \text{Hom}(F, G)$. Next we show that $\text{Hom}(F, G) \subseteq \text{Hom}(F, G')$. Let $h \in \text{Hom}(F, G)$.

As $F \in \mathcal{F}$, for every $u \in V(F)$ there exists $\{v, w\} \in V(F)$ such that $\{uv, vw, wu\} \in E(F)$. As $h : F \to G$ is a homomorphism, we have $\{h(u)h(v), h(v)h(w), h(w)h(u)\} \in E(G)$. This forms a triangle in G. However, the star function S only adds degree-1 vertices to G', resulting in no new triangles. Similarly, for the ring function R with $c > 3$, no new triangles are added. Therefore, this triangle is present in G' making $h \in \text{Hom}(F, G')$. Thus, $\text{Hom}(F, G) \subseteq \text{Hom}(F, G')$ making $\text{Hom}(F, G) = \text{Hom}(F, G')$ and $\hom(F, G') = \hom(F, G)$. As

$$t(F, G) = \frac{\hom(F, G)}{(n+m)^{|V(F)|}} \text{ and } t(F, G') = \frac{\hom(F, G)}{n^{|V(F)|}}$$

we get the result. □

Theorem H.8. *Let $\mathcal{W} = (W, \rho_n, f)$ denote a graphette and let $G' \sim \mathsf{G}(n, W, \rho_n)$ and $G \sim f(G')$ where $f = S$ denotes the star function (GEF 4). Suppose $|V(G')| = n$ and $|V(G)| = n + m$. Then we have the following:*

1. $\hom(\bullet, G) = \hom(\bullet, G') + m$ *and* $t(\bullet, G) = 1$
2. $\hom(\bullet\!\!-\!\!\bullet, G) = \hom(\bullet\!\!-\!\!\bullet, G') + 2m$ *and* $t(\bullet\!\!-\!\!\bullet, G) = \frac{2(|E(G')|+m)}{(n+m)^2} = \frac{n^2}{(n+m)^2}\left(1 + \frac{m}{|E(G')|}\right) t(\bullet\!\!-\!\!\bullet, G')$

Proof. 1. A vertex can be mapped to any vertex in graph G making $\hom(\bullet, G) = |V(G)| = n + m$ and $\hom(\bullet, G') = |V(G')| = n$ giving the first equality. As $t(F, G) = \frac{\hom(F,G)}{|V(G)|^{|V(F)|}}$ we get

$$t(\bullet, G) = \frac{n+m}{|V(G)|} = 1 = \frac{n}{|V(G')|} = t(\bullet, G')\,.$$

2. Adding stars using S adds m vertices and m edges to G'. Thus, $|E(G)| = |E(G')| + m$. An edge can be mapped to any edge in graph G in two ways giving $\hom(\bullet\!\!-\!\!\bullet, G) = 2|E(G)| = 2(|E(G')| + m) = \hom(\bullet\!\!-\!\!\bullet, G') + 2m$. The homomorphism density

$$t(\bullet\!\!-\!\!\bullet, G') = \frac{2|E(G')|}{n^2}\,, \tag{78}$$

$$t(\bullet\!\!-\!\!\bullet, G) = \frac{2|E(G)|}{(n+m)^2} = \frac{2(|E(G')| + m)}{(n+m)^2} \tag{79}$$

By dividing equation (78) from equation (79) we get the result.

□

Theorem H.9. *Let $\mathcal{W} = (W, \rho_n, f)$ denote a graphette and let $G' \sim \mathsf{G}(n, W, \rho_n)$ and $G \sim f(G')$ where $f = R$ denotes the ring function (GEF 3) with ring size $c > 3$ for $c \in \mathcal{C}$. Suppose R adds k rings and $|V(G')| = n$ and $|V(G)| = n + m$. Then we have the following:*

1. $\hom(\bullet, G) = \hom(\bullet, G') + m$ *and* $t(\bullet, G) = 1$
2. $\hom(\bullet\!\!-\!\!\bullet, G) = \hom(\bullet\!\!-\!\!\bullet, G') + 2(m + k)$ *and* $t(\bullet\!\!-\!\!\bullet, G) = \frac{2(|E(G')|+m+k)}{(n+m)^2} = \frac{n^2}{(n+m)^2}\left(1 + \frac{m+k}{|E(G')|}\right) t(\bullet\!\!-\!\!\bullet, G')$

Proof. Part 1 is the same as in Theorem H.8. For part 2 note that a ring has the same number of vertices and edges. If k rings are added to G' and the number of vertices increased by m then the edges would increase by $m + k$ as R connects each ring to a node in G'. The rest is the same as in Theorem H.8. □

I Algorithms

Algorithm 5 BATCHFGW

input Noise graphs $\mathcal{G}^{(0)} = \{G_0^i\}_{i=1}^B$; target graphs $\mathcal{G}^{(1)} = \{G_1^j\}_{j=1}^B$; pretrained encoder f_{enc}; FGW tradeoff $\alpha \in [0, 1]$.

output Cost matrix $D \in \mathbb{R}^{B\times B}$; couplings $T = \{T_{ij}^\star\}_{i,j=1}^B$, one transport plan per pair, indexed identically to D.

Initialize $D \leftarrow \mathbf{0}_{B\times B}$
Initialize $T \leftarrow \emptyset$
for $i = 1$ to B **do**
  for $j = 1$ to B **do**
    $(D_{ij}, T_{ij}) \leftarrow \mathrm{FGW}_\alpha\Big(G_0^i, G_1^j; f_{enc}\Big)$
  end for
end for
return D and T

Algorithm 6 FLOWMATCHINGUPDATE

input Coupled pairs $\mathcal{P} = \{(G_0^i, G_1^{\pi^\star(i)})\}_{i=1}^B$; velocity field parameters θ; regularization parameters $\beta_{\text{end}}, \beta_{\text{val}}, \beta_{\text{atom}}$.

output Updated velocity Field (GNN Transformer) parameters θ.

$\mathcal{L} \leftarrow 0$, $m \leftarrow 0$
for all $(G_0, G_1) \in \mathcal{P}$ **do**
  if $|V(G_0)| = 0$ or $|V(G_1)| = 0$ **then**
    continue
  end if
  $n \leftarrow |V(G_0)| = |V(G_1)|$
  $(A_0, X_0, F_0) \leftarrow \textsc{Graph}(G_0, n)$
  $(A_1, X_1, F_1) \leftarrow \textsc{Graph}(G_1, n)$
  $\tilde{P} \leftarrow \textsc{RowNormalize}(T[G_0, G_1])$ *// soft correspondence from Alg. 5's coupling*
  $A_1 \leftarrow \tilde{P} A_1 \tilde{P}^\top$; $X_1 \leftarrow \tilde{P} X_1$; $F_{1,ab:} \leftarrow \sum_{jk} \tilde{P}_{aj} F_{1,jk:} \tilde{P}_{bk}$ *// move target into source-node order*
  $t \sim \mathcal{U}[0, 1]$
  $A_t \leftarrow (1 - t)A_0 + tA_1$
  $X_t \leftarrow (1 - t)X_0 + tX_1$
  $F_t \leftarrow (1 - t)F_0 + tF_1$
  $\Delta A \leftarrow (A_1 - A_0)$
  $\Delta X \leftarrow (X_1 - X_0)$
  $\Delta F \leftarrow (F_1 - F_0)$
  $(v_A, v_X, v_F) \leftarrow v_\theta(A_t, X_t, F_t, t)$
  $\mathcal{L}_{\text{vel}} = \frac{1}{N_A}\|v_A - \Delta A\|_F^2 + \lambda_x \frac{1}{N_X}\|v_X - \Delta X\|_F^2 + \lambda_e \frac{1}{N_F}\|v_F - \Delta F\|_F^2$
  $\widehat{A}_1 \leftarrow A_t + (1 - t)v_A$
  $\widehat{X}_1 \leftarrow X_t + (1 - t)v_X$
  $\widehat{F}_1 \leftarrow F_t + (1 - t)v_F$
  $\mathcal{L}_{\text{end}} = \frac{1}{N_A}\|\widehat{A}_1 - A_1\|_F^2 + \lambda_x \frac{1}{N_X}\|\widehat{X}_1 - X_1\|_F^2 + \lambda_e \frac{1}{N_F}\|\widehat{F}_1 - F_1\|_F^2$
  $\tilde{\mathcal{L}} \leftarrow \mathcal{L}_{\text{vel}} + \beta_{\text{end}}\mathcal{L}_{\text{end}} + \beta_{\text{val}}\mathcal{L}_{\text{val}} + \beta_{\text{atom}}\mathcal{L}_{\text{atom}}$
  $\mathcal{L} \leftarrow \mathcal{L} + \tilde{\mathcal{L}}$
  $m \leftarrow m + 1$
end for
if $m > 0$ **then**
  $\mathcal{L} \leftarrow \mathcal{L}/m$
  Update θ via AdamW step using $\nabla_\theta \mathcal{L}$ and η
end if
return θ

Algorithm 7 Euler Integration of the Learned Flow

input Velocity field v_θ; initial state (A_0, X_0, F_0); number of steps K; non-negativity activation function $\sigma(\cdot)$.

output Final state (A_1, X_1, F_1).

Set $(A \leftarrow A_0,\ X \leftarrow X_0,\ F \leftarrow F_0)$
$\Delta t \leftarrow 1/K$
for $k = 0$ to $K - 1$ **do**
  $t \leftarrow k/K$
  $(v_A, v_X, v_F) \leftarrow v_\theta(A, X, F, t)$
  $A \leftarrow A + \Delta t\, v_A$
  $X \leftarrow X + \Delta t\, v_X$
  $F \leftarrow F + \Delta t\, v_F$
end for
$A \leftarrow \frac{1}{2}(A + A^\top)$
$A \leftarrow \sigma(A)$
$F \leftarrow \frac{1}{2}(F + F^\top)$
return (A, X, F)

Algorithm 8 Add Disjoint Rings

input Graph $G = (V, E)$; node budget n; ring-size counts $\mathcal{R}' = \{(s, c_s)\}$, where c_s denotes the requested number of cycles of length s.

output Graph $\widetilde{G}$ with $|V(\widetilde{G})| \le n$.

$\widetilde{G} \leftarrow G$
if $\mathcal{R}' = \emptyset$ **then**
  return $\widetilde{G}$
end if
for all $(s, c_s) \in \mathcal{R}'$ **do**
  for $r = 1$ to c_s **do**
    $\texttt{rem} \leftarrow n - |V(\widetilde{G})|$
    if $\texttt{rem} \le 0$ **then**
      break
    end if
    $s' \leftarrow \min(s, \texttt{rem})$
    if $s' < 3$ **then**
      continue
    end if
    $H \leftarrow \mathrm{Cycle}(s')$ *// fresh cycle component*
    $\widetilde{G} \leftarrow \widetilde{G} \uplus H$
    Sample $v \sim \mathrm{Unif}\big(V(\widetilde{G}) \setminus V(H)\big),\quad w \sim \mathrm{Unif}\big(V(H)\big)$
    $E(\widetilde{G}) \leftarrow E(\widetilde{G}) \cup \{(v, w)\}$ *// attach by one edge*
  end for
end for
return $\widetilde{G}$

Algorithm 9 Graphette Sampling with Ring Injection

input Graphon matrix $W \in [0,1]^{m\times m}$; node budget n; sparsity factor ρ (or `None`); stability $\varepsilon > 0$; ring-size counts $\mathcal{R}' = \{(s, c_s)\}$, where c_s is the number of cycles of length s to inject (e.g., $\mathcal{R}' = \{(5,2),(6,1)\}$ requests two 5-cycles and one 6-cycle); max attempts T.

output Undirected graph $G = (V, E)$ with $|V| \le n$.

// Sparsified-graphon sampling (graphex base)

$\bar{W} \leftarrow \frac{1}{m^2}\sum_{a,b=1}^{m} W_{ab}$

if $\rho =$ `None` **then**

  $\rho \leftarrow \frac{1}{\bar{W}\, n} + \varepsilon$

end if

$W' \leftarrow \rho\, W$

Sample $u_1, \ldots, u_n \overset{iid}{\sim} \mathrm{Unif}[0,1]$

Map to indices $\ell_i \leftarrow \min(m-1, \lfloor m u_i \rfloor)$ for $i \in [n]$

Initialize adjacency $A \leftarrow \mathbf{0} \in \{0,1\}^{n\times n}$

for $1 \le i < j \le n$ **do**

  $p_{ij} \leftarrow W'_{\ell_i,\ell_j}$

  Sample $A_{ij} \sim \mathrm{Bernoulli}(p_{ij})$; set $A_{ji} \leftarrow A_{ij}$

end for

$G \leftarrow \textsc{GraphFromAdj}(A)$

// Keep largest connected component, where $CC(G)$ denotes the set of connected components of G

$C^\star \leftarrow \arg\max_{C\in \mathrm{CC}(G)} |C|$

$G \leftarrow G[C^\star]$

if $|V(G)| > n$ **then**

  Sample $S \subseteq V(G)$ uniformly with $|S| = n$; $G \leftarrow G[S]$

end if

// Budgeted disjoint ring injection (attach new components)

$G \leftarrow \textsc{AddDisjointRings}(G, n, \mathcal{R}')$

// Final safety check

if $|V(G)| > n$ **then**

  Sample $S \subseteq V(G)$ uniformly with $|S| = n$; $G \leftarrow G[S]$

end if

return G

I.1 Soft Node Correspondence from FGW Coupling

For each FGW-coupled graph pair (G_0, G_1), let $T^\star \in \mathbb{R}^{n_0\times n_1}$ denote the optimal transport plan obtained from Eq. (2) for this pair (Algorithm 5 returns both the distance D_{ij}, used for Hungarian graph-level pairing, and the corresponding coupling $T^\star$). We row-normalise $T^\star$ to obtain a soft node correspondence $\tilde{P}_{ab} = \frac{T^\star_{ab}}{\sum_{b'} T^\star_{ab'}}$, which is a right-stochastic matrix mapping each target node b's contribution onto source node a; this recovers the exact hard permutation when $T^\star$ is a permutation coupling and degrades gracefully to a soft alignment otherwise. Before constructing any interpolant, the target graph is transported into the source-node ordering:

$$X_1 = \tilde{P} X_1, \qquad A_1 = \tilde{P} A_1 \tilde{P}^\top, \qquad F_{1,ab:} = \sum_{jk} \tilde{P}_{aj} F_{1,jk:} \tilde{P}_{bk}, \tag{80}$$

where the last expression applies $\tilde{P}$ along both node indices of the edge tensor. We then restrict (A_0, X_0, F_0) and (A_1, X_1, F_1) to the common node budget n, ensuring $|V_0| = |V_1| = n$ for every coupled pair, and construct the rectified interpolants in source-node coordinates (Sec. 3.2). This transport step is implemented explicitly in lines 9–10 of Algorithm 6.

J Details of Empirical Evaluation

J.1 Discretisation and chemical validity.

At sampling time, the continuous outputs $(\hat{A}_1, \hat{X}_1, \hat{F}_1)$ obtained after K Euler steps are produced as in Algorithm 7, and are then mapped to a discrete molecular graph as described below. Atom

and bond types are decoded by argmax over their categorical logits, with formal charges decoded analogously for datasets where charge features are used. The adjacency prediction $\hat{A}_1$ is symmetrised and transformed into edge-confidence scores using a sigmoid. These scores are binarised using an adaptive threshold based on a task-specific expected edge budget. Chemical feasibility is then promoted through valence-aware pruning, which iteratively removes the highest-bond-order incident edge from atoms whose current valence exceeds the permitted value, until no such violation remains. Finally, we optionally retain the largest connected component and sanitise the resulting molecule using RDKit.

Post-generation validity correction is also used by existing one-shot molecular graph generators. MoFlow [74] applies an iterative post-hoc valency correction that reduces offending bond orders until the valency constraints are satisfied and retains the largest connected component, while GraphEBM [45] adopts the same correction after Langevin sampling and discretisation. In contrast, the autoregressive GraphAF [57] checks valency during each bond-generation step, rejecting and resampling bond proposals that violate the corresponding constraints. Flowette applies its deterministic terminal projection only after continuous flow integration, leaving the learned velocity field and continuous sampling dynamics unchanged.

J.2 Evaluation Metrics for Synthetic Graph Evaluation

Generation quality is assessed using different metrics: (1) degree distribution (Deg.), (2) clustering coefficient (Clus.), and (3) orbit count (Orbit) statistics. For all metrics, we quantify the discrepancy between the generated and test distributions using Maximum Mean Discrepancy (MMD) [24, 51, 4, 55]. These discrepancies are summarized using the Ratio metric (lower is better). We also report generation quality using validity (V), uniqueness (U), novelty (N), and their combined V.U.N. metric. Following prior work [55], we generate the same number of graphs as in the test set (40 samples) to enable a fair comparison of graph statistic distributions.

We evaluate generative quality by comparing the distributions of key graph statistics computed on generated samples against those of the test set. Specifically, we consider degree distributions, clustering coefficients, and orbit counts. Distributional discrepancies are quantified using the Maximum Mean Discrepancy (MMD). Specifically, we first compute MMD values between the training and test sets for the same statistics, establishing a reference level of distributional shift. We then compute MMD values between the generated and test sets. The Ratio is defined as the average generated:test MMD normalized by the average training:test MMD, providing a scale-free measure of how closely the generated graphs match the test distribution.

Rather than reporting absolute MMD values, we follow a normalised evaluation protocol that reports relative error ratios. For a given statistic, we compute

$$r \;=\; \frac{\mathrm{MMD}^2\big(\mathcal{G}_{\mathrm{gen}},\, \mathcal{G}_{\mathrm{test}}\big)}{\mathrm{MMD}^2\big(\mathcal{G}_{\mathrm{train}},\, \mathcal{G}_{\mathrm{test}}\big)}, \tag{81}$$

where $\mathcal{G}_{\mathrm{gen}}$, $\mathcal{G}_{\mathrm{train}}$, and $\mathcal{G}_{\mathrm{test}}$ denote the generated, training, and test graph sets, respectively. The denominator corresponds to the reference training–test discrepancy reported in prior work [67]. Note that throughout, we use MMD^2, consistent with the convention adopted in DiGress, where the squared MMD is reported.

J.3 Hyperparameter settings for Synthetic Graph Evaluation

For synthetic graph benchmarks, we use a simplified training objective by setting $\beta_{\mathrm{val}} = 0$ and $\beta_{\mathrm{atom}} = 0$, thereby excluding chemistry-specific regularization terms that are not applicable to purely structural graph data. We optimize all models using the AdamW optimizer [48]. For synthetic graph benchmarks, models are trained for 100 epochs with a learning rate of $1e^{-3}$, hidden dimension $H = 128$, and batch size $B = 32$. We set the $\beta_{\mathrm{end}} = 1.0$ and $\lambda_x = \lambda_e = 0.5$, and use an FGW trade-off parameter $\alpha = 0.5$. Euler integration is performed with $K = 50$ steps at sampling time. For the network architecture, we use $L = 3$ attention layers for the Tree and Ego-small datasets, and $L = 4$ attention layers for the SBM dataset.

J.4 Tables of results for synthetic datasets

We compare Flowette against a diverse set of strong baselines across synthetic benchmarks. On the SBM dataset (Table 5), we evaluate against ten methods, including SPECTRE [51], DiGress [67], BwR [18], HSpectre [4], GruM [31], CatFlow [19], DisCo [71], Cometh [59], DeFoG [55], and G2PT [14]. On the Tree dataset (Table 6), we compare against nine baselines, including GRAN [39], DiGress [67], BwR [18], EDGE [13], BiGG [15], GraphGen [23], HSpectre [4], DeFoG [55], and G2PT [14]. On the Ego-small dataset (Table 7), we compare against seven baselines: GraphRNN [73], GRAN [39], GraphCNF [42], GDSS [32], DiscDDPM [25], DiGress [67], and EDGE [13].

SBM Dataset. For SBM graphs in Table 5, our approach outperforms all baselines on Orbit, Validity, Uniqueness, Novelty, and V.U.N., while Degree and Clustering are marginally weaker than the strongest competitors. SBMs are defined by mesoscopic block structure rather than precise local statistics, which is well captured by the graphette prior and reinforced by Flowette. This yields accurate recovery of block-level connectivity and higher-order substructure patterns. Small discrepancies in degree and clustering arise because these second-order marginals can vary within blocks without affecting the overall community structure and are not explicitly constrained by our objective.

Tree Dataset. On the Tree dataset in Table 6, our method achieves the best performance on Orbit, Validity, Uniqueness, Novelty, and the combined V.U.N. score, while slightly underperforming on Degree. This behaviour is consistent with our use of a graphette structural prior and Flowette, which strongly constrain acyclicity and higher-order subgraph structure. As a result, tree-specific substructure patterns and validity are preserved extremely well. In contrast, degree distributions in trees are highly sensitive to structural prior, and our continuous flow prioritizes global structural alignment over exact degree matching, leading to minor deviations in Degree despite structurally correct samples.

Ego-small Dataset. On Ego-small graphs in Table 7, our method achieves the best Orbit score and competitive Degree performance, with Clustering slightly below the top baseline. Ego graphs are dominated by hub-and-spoke and star-like motifs, which align naturally with graphex constructions and are effectively preserved by our flow model. While orbit statistics and degree distributions are recovered, clustering depends on fine-grained triangle closure, which may vary under continuous flow integration even when the overall ego-centric structure is correctly modeled.

Table 5: Generation performance on SBM synthetic graphs.

Model	Deg. ↓	Clus. ↓	Orbit ↓	Valid ↑	Unique ↑	Novel ↑	V.U.N ↑
SPECTRE [51]	0.0015	0.0521	0.0412	52.5	**100.0**	**100.0**	52.5
DiGress [67]	0.0018	0.0485	0.0415	60.0	**100.0**	**100.0**	60.0
BwR [18]	0.0478	0.0638	0.1139	7.5	**100.0**	**100.0**	7.5
HSpectre [4]	0.0141	0.0528	0.0809	75.0	**100.0**	**100.0**	75.0
GruM [31]	0.0015	0.0589	0.0450	-	-	-	85.0
CatFlow [19]	0.0012	0.0498	0.0357	-	-	-	85.0
DisCo [71]	0.0006	0.0266	0.0510	66.2	**100.0**	**100.0**	66.2
Cometh [59]	0.0020	0.0498	0.0383	75.0	**100.0**	**100.0**	75.0
DeFoG [55]	**0.0006**	0.0517	0.0556	90.0	90.0	90.0	90.0
G2PT [14]	0.0035	**0.0120**	0.0007	-	-	-	**100.0**
Flowette (Ours)	0.0076 $\pm$ 0.0035	0.0508 $\pm$ 0.01	**0.0000** $\pm$ 0.0	**100.0** $\pm$ 0.0	**100.0** $\pm$ 0.0	**100.0** $\pm$ 0.0	**100.0** $\pm$ 0.0

Table 6: Generation performance on Tree synthetic graphs.

Model	Deg. ↓	Clus. ↓	Orbit ↓	Valid ↑	Unique ↑	Novel ↑	V.U.N ↑
GRAN [39]	0.1884	0.0080	0.0199	0.0	**100.0**	**100.0**	0.0
DiGress [67]	0.0002	**0.0000**	**0.0000**	90.0	**100.0**	**100.0**	90.0
BwR [18]	0.0016	0.1239	0.0003	0.0	**100.0**	**100.0**	0.0
EDGE [13]	0.2678	**0.0000**	0.7357	0.0	7.5	**100.0**	0.0
BiGG [15]	0.0014	**0.0000**	**0.0000**	**100.0**	87.5	50.0	75.0
GraphGen [23]	0.0105	**0.0000**	**0.0000**	95.0	**100.0**	**100.0**	95.0
HSpectre [4]	0.0004	**0.0000**	**0.0000**	82.5	**100.0**	**100.0**	82.5
DeFoG [55]	**0.0002**	**0.0000**	**0.0000**	96.5	**100.0**	**100.0**	96.5
G2PT [14]	0.0020	**0.0000**	**0.0000**	-	-	-	99.0
Flowette (Ours)	0.0005 $\pm$ 0.002	**0.0000** $\pm$ 0.0	**0.0000** $\pm$ 0.0	**100.0** $\pm$ 0.0	**100.0** $\pm$ 0.0	**100.0** $\pm$ 0.0	**100.0** $\pm$ 0.0

J.5 Evaluation Metrics for Molecular Graph Evaluation

We evaluate generative performance using a collection of complementary molecular metrics [67, 55]. Validity reports the fraction of generated molecules that satisfy basic chemical constraints such as

Table 7: Generation performance on Ego-small synthetic graphs.

Model	Deg. ↓	Clus. ↓	Orbit ↓
GraphRNN [73]	0.0768	1.1456	0.1087
GRAN [39]	0.5778	0.3360	0.0406
GraphCNF [42]	0.1010	0.7654	0.0820
GDSS [32]	0.8189	0.6032	0.3315
DiscDDPM [25]	0.4613	0.1681	0.0633
DiGress [67]	0.0708	0.0092	0.1205
EDGE [13]	**0.0040**	**0.0040**	0.0008
Flowette (Ours)	0.0435 $\pm$ 0.004	0.0098 $\pm$ 0.0025	**0.0000** $\pm$ 0.0

allowable valences. Uniqueness quantifies how many generated molecules are structurally distinct, as determined by non-isomorphic SMILES representations. Novelty measures the proportion of samples that do not appear in the training set. To evaluate distributional similarity, we report the Similarity to Nearest Neighbor (SNN), which captures how close each generated molecule is to its most similar training example using Tanimoto similarity. Scaffold similarity (Scaf.) compares the distributions of Bemis–Murcko core structures. While KL divergence (KL div.) measures discrepancies between generated and reference distributions of selected physicochemical descriptors. Neighborhood Subgraph Pairwise Distance Kernel (NSPDK) evaluates similarity at the level of local graph substructures by comparing distributions of rooted subgraphs and their pairwise distances, thereby capturing higher-order structural consistency between generated and reference molecular graphs.

Let $\mathcal{M}_{\text{gen}} = \{m_i\}_{i=1}^{N}$ denote the set of generated molecules and $\mathcal{M}_{\text{train}}$ the training set.

Validity. Validity measures the fraction of generated molecules that satisfy basic chemical constraints, such as permissible atom valences and bond types. Formally,

$$\text{Validity} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[m_i \text{ is chemically valid}], \tag{82}$$

where $\mathbf{1}[\cdot]$ denotes the indicator function and chemical validity is assessed using RDKit sanitization rules.

Uniqueness. Uniqueness quantifies the proportion of structurally distinct molecules among valid samples, determined via canonical SMILES representations:

$$\text{Uniqueness} = \frac{|\text{Unique}(\mathcal{M}_{\text{gen}}^{\text{valid}})|}{|\mathcal{M}_{\text{gen}}^{\text{valid}}|}. \tag{83}$$

Novelty. Novelty measures the fraction of generated molecules that do not appear in the training set:

$$\text{Novelty} = \frac{1}{|\mathcal{M}_{\text{gen}}^{\text{valid}}|}\sum_{m\in\mathcal{M}_{\text{gen}}^{\text{valid}}} \mathbf{1}[m \notin \mathcal{M}_{\text{train}}]. \tag{84}$$

Similarity to Nearest Neighbour (SNN). SNN measures how close each generated molecule is to its most similar reference molecule in fingerprint space. Let $\tilde{\phi}(\cdot)$ denote a fixed molecular fingerprint map, $\mathcal{M}_{\text{gen}}$ the set of generated molecules, and $\mathcal{M}_{\text{ref}}$ the reference (training) set. We define

$$\text{SNN} = \frac{1}{|\mathcal{M}_{\text{gen}}|}\sum_{m\in\mathcal{M}_{\text{gen}}} \max_{m'\in\mathcal{M}_{\text{ref}}} \text{Tanimoto}\big(\tilde{\phi}(m), \tilde{\phi}(m')\big), \tag{85}$$

where the Tanimoto similarity is computed over binary fingerprints and the maximum is taken over all reference molecules.

Scaffold Similarity

Scaffold similarity (Scaf.). Scaffold similarity evaluates how well the distribution of molecular scaffolds is preserved in generated samples. Let $\mathcal{S}$ denote the union of Bemis–Murcko scaffolds observed in the reference and generated sets. We construct scaffold count vectors

$$\mathbf{r} = (r_s)_{s\in\mathcal{S}}, \qquad \mathbf{g} = (g_s)_{s\in\mathcal{S}},$$

where r_s and g_s denote the number of molecules containing scaffold s in the reference and generated sets, respectively (with missing entries treated as zero) [54]. Scaffold similarity is defined as the cosine similarity between these count vectors:

$$\text{Scaf.} = \frac{\langle \mathbf{r}, \mathbf{g} \rangle}{\|\mathbf{r}\|_2 \, \|\mathbf{g}\|_2}. \tag{86}$$

This metric captures agreement between scaffold frequency distributions while being invariant to absolute dataset size.

Physicochemical KL divergence. To assess distributional fidelity at the level of physicochemical properties, we follow standard GuacaMol-style benchmarks and compare descriptor distributions between training and generated molecules. Let $\mathcal{D}$ denote a fixed set of scalar descriptors, e.g. Bertz complexity, logP, molecular weight, topological polar surface area, and several integer-valued counts (H-bond donors/acceptors, rotatable bonds, aliphatic and aromatic rings) [8]. For each descriptor $j \in \mathcal{D}$ and molecule x, we write $d_j(x) \in \mathbb{R}$ for its value.

Given a reference (training) set $\mathcal{X}_{\text{ref}}$ and a generated set $\mathcal{X}_{\text{gen}}$, we construct empirical distributions of d_j under each set. For integer-valued descriptors, we estimate discrete histograms

$$p^{(j)}_{\text{ref}}(k) \;=\; \frac{1}{|\mathcal{X}_{\text{ref}}|} \sum_{x \in \mathcal{X}_{\text{ref}}} \mathbf{1}\{d_j(x) = k\}, \qquad p^{(j)}_{\text{gen}}(k) \;=\; \frac{1}{|\mathcal{X}_{\text{gen}}|} \sum_{x \in \mathcal{X}_{\text{gen}}} \mathbf{1}\{d_j(x) = k\},$$

where k ranges over a finite set of bins. The KL divergence for descriptor j is then

$$D^{(j)}_{\text{KL}} \;=\; \sum_k p^{(j)}_{\text{ref}}(k) \log \frac{p^{(j)}_{\text{ref}}(k)}{p^{(j)}_{\text{gen}}(k)}. \tag{87}$$

For continuous-valued descriptors, we estimate densities $p^{(j)}_{\text{ref}}(z)$ and $p^{(j)}_{\text{gen}}(z)$ (e.g. via kernel density estimation or smoothed histograms) and define

$$D^{(j)}_{\text{KL}} \;=\; \int p^{(j)}_{\text{ref}}(z) \log \frac{p^{(j)}_{\text{ref}}(z)}{p^{(j)}_{\text{gen}}(z)} \, dz, \tag{88}$$

which in practice is approximated numerically.

Aggregate KL score (KL div.). The overall physicochemical KL metric aggregates these divergences across all descriptors and the internal-similarity distribution. Let $\mathcal{J}$ denote the index set consisting of all descriptor dimensions together with the internal-similarity term. We report either the average KL,

$$\text{KL}_{\text{physchem}} \;=\; \frac{1}{|\mathcal{J}|} \sum_{j \in \mathcal{J}} D^{(j)}_{\text{KL}}, \tag{89}$$

or the corresponding similarity score

$$\text{Score}_{\text{physchem}} \;=\; \frac{1}{|\mathcal{J}|} \sum_{j \in \mathcal{J}} \exp\!\big(-D^{(j)}_{\text{KL}}\big), \tag{90}$$

where smaller KL values (or larger scores) indicate closer agreement between the generated and training distributions of physicochemical descriptors and internal similarities.

Neighborhood Subgraph Pairwise Distance Kernel (NSPDK). NSPDK evaluates similarity at the level of local graph substructures by comparing distributions of rooted subgraphs and their pairwise distances. Given feature maps $\phi(\cdot)$ induced by the kernel, the similarity between generated and reference sets is measured via a maximum mean discrepancy (MMD):

$$\text{NSPDK} = \left\| \frac{1}{|\mathcal{M}_{\text{gen}}|} \sum_{m \in \mathcal{M}_{\text{gen}}} \phi(m) - \frac{1}{|\mathcal{M}_{\text{train}}|} \sum_{m \in \mathcal{M}_{\text{train}}} \phi(m) \right\|^2_2. \tag{91}$$

NSPDK captures higher-order structural consistency beyond local degree statistics, making it particularly sensitive to discrepancies in functional groups and subgraph arrangements.

Table 8: Runtime comparison between Flowette (under the default online coupling protocol; see Sec. J.8 for amortised alternatives) and DeFoG. Flowette training time and pairing time (FGW + Hungarian) are reported separately to highlight the marginal overhead of coupling.

Dataset	DeFoG Model Train (h)	Flowette Pairing (h)	Flowette Model Train (h)	DeFoG Sample (h)	Flowette Sample (h)	Graphs Sampled
Tree	8.0	0.016	8.50	0.07	0.04	40
QM9	6.5	0.560	6.90	0.20	0.16	10,000
ZINC250K	14.0	0.750	14.60	4.80	3.70	10,000
MOSES	46.0	1.680	46.50	5.00	4.30	25,000

J.6 Hyperparameter setup for molecular graph generation

We compare our approach against seven baseline models on the Guacamol and MOSES benchmarks: Digress [67], DisCo [71], Cometh [59], GraphBFN [60], G2PT [14], GEEL [28], and DeFoG [55]. We further evaluate our method on the QM9 and ZINC250K datasets using ten comparable baselines: EDP-GNN [53], GraphARM [37], DiGress [67], GDSS [32], GDSS + TF [32], GruM [31], SID [5], DeFoG [55], GraphBFN [60], and GBD [47].

For molecular graph benchmarks, we optimize all models using the training objective defined in Eq. (6) and employ the AdamW optimizer [48]. All models are trained for 500 epochs with a learning rate of $1e^{-3}$, hidden dimension $H = 128$, batch size $B = 32$, and loss weights $\lambda_x = \lambda_e = 0.5$. We set the FGW trade-off parameter to $\alpha = 0.5$. For the QM9, ZINC250K, and Guacamol datasets, we use $\beta_{\text{val}} = 0.5$, $\beta_{\text{atom}} = 0.5$, and $\beta_{\text{end}} = 1.0$. For the MOSES dataset, we set $\beta_{\text{val}} = 0.3$, $\beta_{\text{atom}} = 0.4$, and $\beta_{\text{end}} = 0.8$. During sampling, Euler integration is performed with $K = 200$ steps. Regarding model architecture, we use $L = 5$ attention layers for QM9 and ZINC250K, and $L = 4$ attention layers for GuacaMol and MOSES. Following existing evaluation protocols [55], we generate 10000 graphs for QM9, ZINC250K, and Guacamol, and 25000 graphs for MOSES at inference time to ensure fair comparison.

J.7 Runtime Analysis

We evaluate the computational efficiency of Flowette under the same experimental setup as DeFoG [55], using batch size $B = 512$ and 1000 training epochs. As shown in Table 8, Flowette remains competitive in total training time (i.e., FGW + Hungarian pairing + model training) across all benchmarks. The additional FGW + Hungarian pairing overhead is small, ranging from 0.016h (Tree) to 1.68h (MOSES), and constitutes only a minor fraction of the overall training cost. Moreover, this overhead can be further reduced, as pairings can be pre-computed offline once before training and reused across epochs. However, Flowette is strictly faster than DeFoG at inference across all benchmarks. Since FGW coupling and Hungarian matching are training-only operations, inference requires only standard Euler integration of the learned velocity field, with no CTMC simulation and no rate matrix computation, that DeFoG requires at sampling time.

J.8 Scalability of Coupling Protocols

Coupling protocols. The default protocol, used throughout Algorithm 1 and all main results unless stated otherwise, is *online coupling*: FGW costs and the Hungarian assignment are recomputed for each minibatch at every epoch, so no fixed precomputed source pool is assumed. We additionally support two optional amortised variants for settings where repeated FGW computation is a bottleneck: *fixed-pool* coupling, which performs one blockwise pass over the training set before the first epoch, samples one source graph per target, and caches the resulting source graph and transport plan keyed by the target's dataset index for reuse across all subsequent epochs; and *periodic* coupling, which refreshes this cache every $\tilde{\mathcal{R}}$ epochs. In both amortised variants, precomputation is performed in minibatch-sized $B \times B$ blocks rather than as a single dataset-wide matching problem, and the cached correspondence remains valid under shuffled training since it is indexed by target graph identity rather than minibatch position.

To isolate the cost of the coupling step itself and evaluate scalability with graph size, we conduct a controlled scaling study on Tree and SBM graphs with $n \in \{100, 500, 1000\}$ nodes, using 2000 training, 500 validation, and 500 test graphs per setting. Trees are sampled from random Prüfer sequences; four-community SBMs use $p_{\text{in}} = 30/n$ and $p_{\text{out}} = 4/n$ to hold expected degree approximately constant across n. We compare the three coupling protocols – *online, fixed-pool, and*

periodic ($\tilde{\mathcal{R}} = 10$), under identical hardware, model settings, and a 1000-epoch training budget, reporting total end-to-end training time (FGW coupling + Hungarian matching + model training).

Table 9: Total end-to-end training time (hours) under online, fixed-pool, and periodic ($\tilde{\mathcal{R}} = 10$) coupling, across graph size n and dataset family.

Dataset	n	Online (h)	Fixed-pool (h)	Periodic, $\tilde{\mathcal{R}}$=10 (h)
Tree	100	8.90	3.50	5.15
	500	16.25	5.86	9.45
	1000	26.30	8.67	12.36
SBM	100	12.25	5.65	7.91
	500	22.05	8.82	12.75
	1000	34.56	11.58	14.95

Repeatedly recomputing FGW under the online protocol introduces substantial training overhead as n grows, whereas amortising the coupling considerably improves practical scalability. Across both Tree and SBM graphs, fixed-pool coupling reduces end-to-end training time by a large margin relative to online coupling, while periodic coupling with $\tilde{\mathcal{R}} = 10$ offers a smaller but still meaningful reduction, consistent with its role as an intermediate trade-off between full re-coupling and a static correspondence. This consistent trend across sparse, acyclic Trees and community-structured SBMs indicates that the benefit is not specific to a particular graph family. Fixed-pool coupling gives the lowest runtime, while periodic coupling offers an intermediate trade-off by periodically refreshing the alignment during training. Neither FGW nor Hungarian matching is required at inference under any of the three protocols, so this overhead is confined entirely to training.

J.9 Ablation Analysis

J.9.1 Ablation Analysis of Regularization Terms

The comparatively poor molecular performance obtained with $\mathcal{L}_{\text{vel}}$ alone (Table 10) should not be interpreted as evidence of insufficient backbone capacity. Rather, it exposes a mismatch between per-entry MSE-based continuous regression and discrete chemical feasibility: after endpoint discretization, a small error in an atom- or bond-type prediction can change a categorical assignment and violate valence, invalidating the entire molecule even when its contribution to the continuous regression loss is small.

The endpoint and chemistry-aware terms address complementary aspects of this mismatch. As shown in Sec. 3.3 and App. E.3.1, $\mathcal{L}_{\text{end}}$ does not introduce an independent supervision target; under the rectified parameterization it is exactly a $(1-t)^2$ temporal reweighting of the same velocity residual penalized by $\mathcal{L}_{\text{vel}}$. The factor follows directly from the remaining extrapolation horizon in the one-step endpoint estimate, so the same velocity residual induces a larger predicted endpoint error when evaluated earlier in the trajectory. Consequently, combining $\mathcal{L}_{\text{vel}}$ with $\beta_{\text{end}}\mathcal{L}_{\text{end}}$ provides an endpoint-aware temporal reweighting that places greater emphasis on residuals with a larger effect on the predicted terminal state. In contrast, $\mathcal{L}_{\text{val}}$ directly discourages chemically infeasible bonding, while $\mathcal{L}_{\text{atom}}$ stabilizes the predicted atom-type composition. These terms provide complementary, low-overhead supervision without changing the FGW coupling, velocity-field architecture, or underlying continuous transport process.

To evaluate the role of each component in the Flowette training objective, we conduct an ablation study by selectively removing its key elements as shown in Table 10.

- **NoReg/ only** $\mathcal{L}_{vel}$: This variant removes all regularization terms and only keeps $\mathcal{L}_{vel}$.
- **NoValenceReg/ no** $\mathcal{L}_{val}$: This variant removes only the regularization term $\mathcal{L}_{val}$.
- **NoAtomReg/ no** $\mathcal{L}_{atom}$: This variant removes only the regularization term $\mathcal{L}_{atom}$.
- **NoEndReg/ no** $\mathcal{L}_{end}$: This variant removes only the regularization term $\mathcal{L}_{end}$.
- **NoChemReg/ only** $\mathcal{L}_{vel}$ **and** $\mathcal{L}_{end}$: This variant removes both chemistry-aware regularization terms $\mathcal{L}_{val}$ and $\mathcal{L}_{atom}$.

Table 10: Ablation study of graph generation performance of QM9 and ZINC250K. Recall from Equation (6), the components in the training objective: the velocity loss $\mathcal{L}_{\text{vel}}$, endpoint consistency loss $\mathcal{L}\text{end}$, soft valence constraint $\mathcal{L}_{\text{val}}$, and atom-type matching $\mathcal{L}_{\text{atom}}$.

	QM9			ZINC250K		
Variant	Valid	Unique	Novel	Valid	Unique	Novel
only $\mathcal{L}_{\text{vel}}$	69.0 $\pm$ 0.10	65.0 $\pm$ 0.02	65.0 $\pm$ 0.06	75.0 $\pm$ 0.05	83.1 $\pm$ 0.10	83.1 $\pm$ 0.03
no $\mathcal{L}_{\text{val}}$	76.0 $\pm$ 0.07	90.7 $\pm$ 0.04	90.7 $\pm$ 0.06	88.0 $\pm$ 0.10	91.0 $\pm$ 0.05	91.0 $\pm$ 0.04
no $\mathcal{L}_{\text{atom}}$	79.0 $\pm$ 0.05	93.6 $\pm$ 0.06	93.6 $\pm$ 0.03	92.1 $\pm$ 0.02	95.0 $\pm$ 0.08	95.0 $\pm$ 0.06
no $\mathcal{L}_{\text{end}}$	71.0 $\pm$ 0.06	79.0 $\pm$ 0.05	79.0 $\pm$ 0.05	79.0 $\pm$ 0.11	85.0 $\pm$ 0.04	85.0 $\pm$ 0.07
only $\mathcal{L}_{\text{vel}}$ and $\mathcal{L}_{\text{end}}$	73.1 $\pm$ 0.05	85.0 $\pm$ 0.06	85.0 $\pm$ 0.05	82.6 $\pm$ 0.03	90.5 $\pm$ 0.04	90.5 $\pm$ 0.04
Full model	**99.8** $\pm$ 0.09	**99.3** $\pm$ 0.05	**99.4** $\pm$ 0.06	**99.9** $\pm$ 0.10	**100.0** $\pm$ 0.0	**100.0** $\pm$ 0.0

The chemistry-aware terms $\mathcal{L}_{\text{val}}$ and $\mathcal{L}_{\text{atom}}$ primarily govern feasibility and diversity. Following endpoint consistency, removing valence supervision (**no** $\mathcal{L}_{\text{val}}$) causes a pronounced validity drop on both datasets, demonstrating the importance of soft valence constraints for preventing over-bonding. In contrast, removing atom-type marginal matching (**no** $\mathcal{L}_{\text{atom}}$) has a milder effect on validity, uniqueness and novelty. Removing both terms (**only** $\mathcal{L}_{\text{vel}}$ **and** $\mathcal{L}_{\text{end}}$) yields intermediate performance, confirming that the two regularizers provide complementary signals. Velocity matching alone is insufficient, and strong performance emerges only when all components are jointly enforced.

Why velocity matching alone is insufficient on molecules. The performance of the $\mathcal{L}_{\text{vel}}$-only variant reflects the interaction between continuous regression and discrete molecular validity, rather than an architectural inability to represent the transport field. The velocity objective penalizes numerical residuals in the continuous graph representation, whereas chemical validity is determined after categorical decoding and is highly non-smooth. For example, a small perturbation of bond-type logits may change the decoded bond from single to double and cause an atom to exceed its allowed valence. Such an error can invalidate the entire molecule even when it contributes only weakly to the aggregate regression loss. Conversely, a residual of comparable magnitude can leave the decoded molecular graph unchanged. Thus, continuous velocity error is not uniformly aligned with the downstream chemical consequence of that error.

This issue is specific to domains with hard combinatorial constraints, such as valence, and does not indicate that the overall Flowette backbone is molecule-specific: the same backbone – FGW-coupled supervision, the permutation-equivariant GNN transformer, and the rectified flow-matching objective – achieves state-of-the-art results on our synthetic benchmarks (Tree, SBM, Ego-small; Sec. 5.2) with $\beta_{\text{val}} = \beta_{\text{atom}} = 0$ and no chemistry terms whatsoever. The backbone is therefore already domain-general and demonstrably strong on its own; it is only when the target domain imposes hard combinatorial constraints that additional supervision is needed to align the training signal with what validity means in that domain.

Endpoint consistency. Removing $\mathcal{L}_{\text{end}}$ substantially degrades validity and uniqueness on both molecular datasets. As shown in App. E.3.1, $\mathcal{L}_{\text{end}}$ is not an independent endpoint supervision target; it is exactly a $(1-t)^2$ reweighting of the velocity residual already penalized by $\mathcal{L}_{\text{vel}}$. This endpoint-aware temporal reweighting places greater emphasis on velocity residuals with a larger effect on the predicted terminal state, corresponding to earlier points along the trajectory with a longer remaining extrapolation horizon. The ablation therefore indicates that this non-uniform temporal weighting contributes materially to molecular generation quality, even though it introduces no additional transport target.

Chemistry-aware endpoint regularization. The terms $\mathcal{L}_{\text{val}}$ and $\mathcal{L}_{\text{atom}}$ supply targeted supervision for chemical consequences that are weakly represented by the per-entry MSE-based continuous regression objective. Removing $\mathcal{L}_{\text{val}}$ causes a pronounced validity drop on both QM9 and ZINC250K, confirming that explicit soft valence supervision helps prevent over-bonding. Removing $\mathcal{L}_{\text{atom}}$ produces a milder degradation, consistent with its role in stabilizing global atom-type composition rather than directly imposing a hard local chemical constraint. Removing both chemistry-aware terms yields intermediate performance. These regularizers require only coarse domain information–allowable valences and target atom-type marginals–and operate on the predicted endpoint. They do not alter the FGW coupling, the velocity-field architecture, or the continuous transport process.

Table 11: Sensitivity analysis of the pre-trained encoder on QM9 and ZINC250K datasets.

Variant	QM9			ZINC250K		
	Valid	Unique	NSPDK	Valid	Unique	NSPDK
Flowette (pretrained GIN encoder)	**99.81** $\pm$ **0.09**	**99.30** $\pm$ **0.05**	**0.0003** $\pm$ **0.0002**	**99.90** $\pm$ **0.10**	**100.00** $\pm$ **0.0**	**0.0006** $\pm$ **0.0004**
Flowette (1-WL features)	98.30 $\pm$ 0.05	98.00 $\pm$ 0.06	0.0008 $\pm$ 0.0003	98.70 $\pm$ 0.030	99.00 $\pm$ 0.050	0.0018 $\pm$ 0.0007
Flowette (Early stopped)	94.30 $\pm$ 0.06	93.00 $\pm$ 0.04	0.0153 $\pm$ 0.0008	93.80 $\pm$ 0.060	92.60 $\pm$ 0.030	0.0316 $\pm$ 0.0003
Flowette (Random encoder)	88.30 $\pm$ 0.08	87.00 $\pm$ 0.06	0.0656 $\pm$ 0.0006	88.80 $\pm$ 0.070	88.60 $\pm$ 0.050	0.0819 $\pm$ 0.0006

Table 12: Ablation study: Graphette prior vs. standard graphon on synthetic datasets.

Dataset	Model Variant	Deg $\downarrow$	Clus. $\downarrow$	Orbit $\downarrow$	Valid $\uparrow$	Unique $\uparrow$	Novel $\uparrow$	V.U.N $\uparrow$
Tree	Flowette (graphette prior)	**0.00005** $\pm$ **0.002**	**0.0000** $\pm$ **0.0**	**0.0000** $\pm$ **0.0**	**100.0** $\pm$ **0.0**	**100.0** $\pm$ **0.0**	**100.0** $\pm$ **0.0**	**100.0** $\pm$ **0.0**
	Flowette (standard graphon)	0.0472 $\pm$ 0.008	0.0136 $\pm$ 0.001	0.0012 $\pm$ 0.005	75.5 $\pm$ 0.03	90.0 $\pm$ 0.02	100.0 $\pm$ 0.0	67.95 $\pm$ 0.06
SBM	Flowette (graphette prior)	**0.0076** $\pm$ **0.003**	**0.0508** $\pm$ **0.01**	**0.0000** $\pm$ **0.0**	**100.0** $\pm$ **0.0**	**100.0** $\pm$ **0.0**	**100.0** $\pm$ **0.0**	**100.0** $\pm$ **0.0**
	Flowette (standard graphon)	0.1207 $\pm$ 0.002	0.8515 $\pm$ 0.003	0.0105 $\pm$ 0.004	80.0 $\pm$ 0.02	90.5 $\pm$ 0.03	85.5 $\pm$ 0.04	61.90 $\pm$ 0.03

Table 10 therefore shows that they are targeted corrections for the mismatch between continuous MSE-based training errors and discrete chemical feasibility, rather than components of a molecule-specific redesign.

J.9.2 Sensitivity Analysis of the Pretrained Encoder f_{enc}

Pre-trained encoder. We use a GIN encoder (3-layer GINEConv, hidden size 128, output size 64, learning rate 1e-3, batch-size 32, and epochs 50) trained once in a self-supervised manner as a graph autoencoder. It is optimized using cross-entropy losses to reconstruct atom and bond types from node and edge features, without external labels. The encoder is trained only on the training split of each dataset and is never exposed to validation or test data, preventing data leakage. After training, it is frozen and used as a fixed structural embedding extractor, receiving no gradient updates during FM training. Its role is limited to computing node embeddings for FGW distance computation within mini-batches; it does not influence the velocity field, integration process, or generated outputs.

To evaluate sensitivity to encoder quality, we conduct a four-condition ablation, as shown in Table 11: (1) GIN encoder (50 epochs, fully pretrained), (2) training-free 1-WL (Weisfeiler-Lehman [56]) features (3 iterations of WL, zero parameters), (3) Early stopped GIN encoder (5 epochs, partially trained), and (4) a random encoder (weights fixed with default initialisation, zero training). Notably, the gap between the pretrained GIN and the training-free 1-WL baseline is consistently the smallest across all conditions, metrics, and datasets, while performance degrades progressively as encoder quality decreases from 1-WL to early-stopped to random. This confirms that structural information is far more critical than learned representations in our framework.

J.9.3 Ablation of Graphette Prior vs. Standard Graphon Prior

We address this via a controlled ablation comparing Flowette with the graphette prior to a variant using a standard graphon prior, with FGW coupling and all loss terms fixed. As shown in Tables 12 and 13, replacing the graphette prior causes large and consistent degradation. On Tree, Deg increases by 0.04715 and V.U.N decreases by 32.05. On SBM, Clus. increases by 0.8007 and V.U.N decreases by 38.10. On QM9, Valid decreases by 24.31, Unique by 30.40, and NSPDK increases by 0.0757. On ZINC250K, Valid decreases by 19.90, Unique by 15.00, and NSPDK increases by 0.0554. These consistent changes confirm that the graphette prior is a critical independent component for capturing the dominant topological properties of each graph family.

J.9.4 Ablation of FGW coupling vs. Alternative Pairings

We provide a controlled ablation comparing FGW coupling against independent (random) pairing and Euclidean OT, with the graphette prior and all loss terms held fixed. As shown in Tables 14 and 15, FGW consistently outperforms both alternatives across all datasets and metrics. On Tree, Deg increases by 0.00802 (random) and 0.00073 (Euclidean OT), while V.U.N decreases by 18.55 and 9.28, respectively. On SBM, Clus. increases by 0.4251 (random) and 0.0424 (Euclidean OT), while V.U.N decreases by 19.23 and 10.99. On QM9, Valid decreases by 9.78 (random) and 5.31 (Euclidean OT), Unique by 9.65 and 6.05, and NSPDK increases by 0.0177 and 0.0062. On ZINC250K, Valid decreases by 10.40 (random) and 5.90 (Euclidean OT), Unique by 8.50 and 4.50, and NSPDK increases by 0.0154 and 0.0024. These consistent degradations confirm that FGW

Table 13: Ablation study: Graphette prior vs. standard graphon on molecular datasets.

Dataset	Model Variant	Valid ↑	Unique ↑	NSPDK ↓
QM9	Flowette (graphette prior)	**99.81** ± **0.09**	**99.30** ± **0.05**	**0.0003** ± **0.0002**
	Flowette (standard graphon)	75.50 ± 0.03	68.90 ± 0.05	0.0760 ± 0.0006
ZINC250K	Flowette (graphette prior)	**99.90** ± **0.10**	**100.00** ± **0.0**	**0.0006** ± **0.0004**
	Flowette (standard graphon)	80.00 ± 0.05	85.00 ± 0.03	0.0560 ± 0.0005

Table 14: Ablation study: FGW coupling vs. alternative pairings on synthetic datasets.

Dataset	Model Variant	Deg ↓	Clus. ↓	Orbit ↓	Valid ↑	Unique ↑	Novel ↑	V.U.N ↑
Tree	Flowette (FGW)	**0.00005** ± **0.002**	**0.0000** ± **0.0**	**0.0000** ± **0.0**	**100.0** ± **0.0**	**100.0** ± **0.0**	**100.0** ± **0.0**	**100.0** ± **0.0**
	Flowette (random)	0.00807 ± 0.005	0.0055 ± 0.003	0.0000 ± 0.0	90.50 ± 0.06	90.0 ± 0.02	100.0 ± 0.0	81.45 ± 0.04
	Flowette (Euclidean OT)	0.00078 ± 0.003	0.0003 ± 0.005	0.0000 ± 0.0	95.50 ± 0.05	95.0 ± 0.04	100.0 ± 0.0	90.72 ± 0.06
SBM	Flowette (FGW)	**0.0076** ± **0.003**	**0.0508** ± **0.01**	**0.0000** ± **0.0**	**100.0** ± **0.0**	**100.0** ± **0.0**	**100.0** ± **0.0**	**100.0** ± **0.0**
	Flowette (random)	0.0151 ± 0.003	0.4759 ± 0.001	0.0012 ± 0.001	89.50 ± 0.03	90.25 ± 0.05	100.0 ± 0.0	80.77 ± 0.05
	Flowette (Euclidean OT)	0.0091 ± 0.002	0.0932 ± 0.005	0.0005 ± 0.004	93.50 ± 0.06	95.20 ± 0.02	100.0 ± 0.0	89.01 ± 0.03

coupling is a critical independent component, producing more coherent and reliable supervision than both feature-agnostic random pairing and feature-only Euclidean OT.

J.9.5 Ablation of FGW vs. Standard Gromov-Wasserstein Coupling

Standard GW coupling is recovered from FGW (Eq. (3)) as a special case: setting $\alpha = 1$ removes the feature term entirely, and even at $\alpha < 1$ the feature term becomes non-discriminative when node features are absent or constant. Our motivation for using FGW throughout is therefore not to outperform GW on purely structural datasets, but to provide a single coupling mechanism that also handles attributed graphs: for molecules, topology-only matching can align structurally similar but chemically incompatible atoms, introducing noisy feature interpolation and velocity targets, which FGW resolves by jointly matching topology and atom identity.

To isolate the feature contribution, we compare FGW ($\alpha = 0.5$) against GW ($\alpha = 1.0$) on the non-attributed Tree and SBM benchmarks, holding the graphette prior, velocity-field architecture, and training objective fixed (Table 16). As expected, the two variants perform essentially identically: both achieve 100% validity, uniqueness, novelty, and V.U.N., with only negligible differences in structural MMD metrics. This confirms that FGW reduces to GW when attributes are uninformative, while retaining the flexibility needed for attributed molecular graphs (Sec. 3.1).

J.9.6 Ablation of Pre- vs. Post-Projection Validity

The terminal projection described in Sec. J.1 is a deterministic, inference-only step, so its contribution to reported validity can be isolated from that of the learned dynamics by evaluating the same generated trajectories before and after projection is applied. Table 17 reports this comparison on QM9, ZINC250K, and MOSES. Pre-projection outputs already achieve high validity (98.06% on QM9, 98.25% on ZINC250K, 97.95% on MOSES), indicating that the learned flow itself produces predominantly chemically feasible structures. Projection provides a modest additional gain of 1.75, 1.65, and 1.10 percentage points respectively, rather than converting largely invalid outputs into valid molecules. Uniqueness and novelty shift only moderately, and scaffold scores remain close between the two settings, indicating that projection corrects residual local valence violations without substantially redefining molecular identity or global structure. This is consistent with Flowette's training objective: the differentiable valence regulariser $\mathcal{L}_{\text{val}}$ already encourages chemically feasible endpoints during learning (Sec. 3.3), while terminal projection acts as a final safeguard against residual discretisation or finite-step integration error rather than as the primary source of validity.

Post-generation validity correction of this kind is not unique to Flowette: MoFlow [74] applies post-hoc validity correction when assembling generated molecular graphs, GraphEBM [45] applies validity correction to generated node and adjacency tensors, and autoregressive methods such as GraphAF [57] enforce validity checks during generation. We therefore report Flowette's pre-projection metrics above as a direct, projection-free comparison point, and its post-projection metrics separately (Tables 2 and 3) as the performance of the complete generation pipeline. We note that applying an identical projection to a baseline is meaningful only when that baseline exposes comparable atom- and bond-level confidence scores; absent such scores, confidence-based bond correction would require an arbitrary ranking over discrete outputs and would not constitute an equivalent procedure. For baselines

Table 15: Ablation study: FGW coupling vs. alternative pairings on molecular datasets.

Dataset	Model Variant	Valid ↑	Unique ↑	NSPDK ↓
QM9	Flowette (FGW)	**99.81** ± **0.09**	**99.30** ± **0.05**	**0.0003** ± **0.0002**
	Flowette (random)	90.03 ± 0.01	89.65 ± 0.04	0.0180 ± 0.0005
	Flowette (Euclidean OT)	94.50 ± 0.03	93.25 ± 0.02	0.0065 ± 0.0006
ZINC250K	Flowette (FGW)	**99.90** ± **0.10**	**100.00** ± **0.0**	**0.0006** ± **0.0004**
	Flowette (random)	89.50 ± 0.07	91.50 ± 0.05	0.0160 ± 0.0003
	Flowette (Euclidean OT)	94.00 ± 0.05	95.50 ± 0.03	0.0030 ± 0.0004

Table 16: FGW ($\alpha = 0.5$) vs. standard GW ($\alpha = 1.0$) coupling on non-attributed synthetic benchmarks.

Dataset	Coupling	Deg. ↓	Clus. ↓	Orbit ↓	Valid ↑	Unique ↑	Novel ↑	V.U.N ↑
Tree	FGW (α=0.5)	0.00005 ± 0.002	0.0000 ± 0.0	0.0000 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
	GW (α=1.0)	0.00010 ± 0.003	0.0000 ± 0.0	0.0000 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
SBM	FGW (α=0.5)	0.0076 ± 0.003	0.0508 ± 0.01	0.0000 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0
	GW (α=1.0)	0.0080 ± 0.005	0.0595 ± 0.03	0.0000 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0	100.0 ± 0.0

that do expose compatible logits or probabilities, the same projection could be applied and evaluated directly.

K Observed Limitations

While Flowette shows strong performance, several aspects offer avenues for further improvement: (i) online FGW coupling incurs $\mathcal{O}(B^2n^3)$ precomputation cost (Table 8), which may limit scalability and motivates more efficient couplings; (ii) graphette specification relies on coarse domain knowledge (e.g., rings, hubs, trees); our controlled ablations (Sec. 5.4) show this choice matters – replacing the intended edit function with an incompatible or contradictory one substantially degrades validity and NSPDK on QM9 and ZINC250K, so automating f via graphon-estimation methods [69, 3] is a promising direction rather than a purely cosmetic one; (iii) the optional endpoint regularizers $\mathcal{L}_{\text{val}}$ and $\mathcal{L}_{\text{atom}}$ encode coarse molecular constraints; analogous endpoint constraints may be required in other domains with hard combinatorial validity conditions, although Flowette achieves strong synthetic performance without these chemistry-aware terms (100% V.U.N. on Tree and SBM); (iv) the continuous relaxation requires a discrete projection step at generation time, which may introduce minor artefacts and motivates tighter integration.

Table 17: Validity, uniqueness, novelty, and scaffold similarity before and after valence-aware terminal projection. Pre-projection metrics isolate the contribution of the learned dynamics; post-projection metrics reflect the full generation pipeline reported in Tables 2 and 3.

Dataset	Variant	Valid	Unique	Novel	Scaff.
QM9	Pre-projection	98.06 ± 0.06	98.28 ± 0.07	98.00 ± 0.04	11.30
	Post-projection	**99.81** ± 0.09	**99.30** ± 0.05	**99.40** ± 0.06	**13.60**
ZINC250K	Pre-projection	98.25 ± 0.02	98.35 ± 0.04	98.56 ± 0.05	14.10
	Post-projection	**99.90** ± 0.10	**100.00** ± 0.0	**100.00** ± 0.0	**15.45**
MOSES	Pre-projection	97.95 ± 0.05	98.00 ± 0.03	97.06 ± 0.05	14.45
	Post-projection	**99.05** ± 0.0	**99.90** ± 0.0	**98.10** ± 0.0	**15.30**

L Graphs generated by Flowette

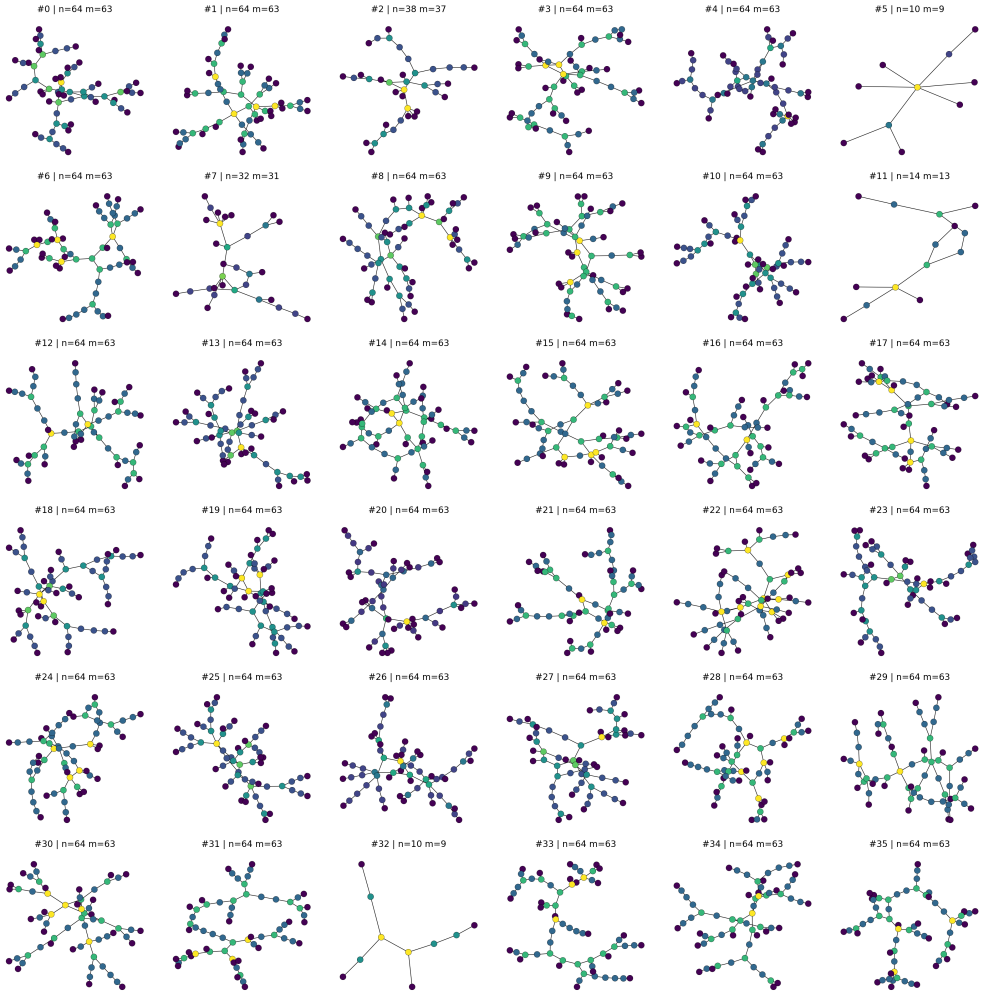

Figure 7: Graphs generated by Flowette for the Tree synthetic datasets.

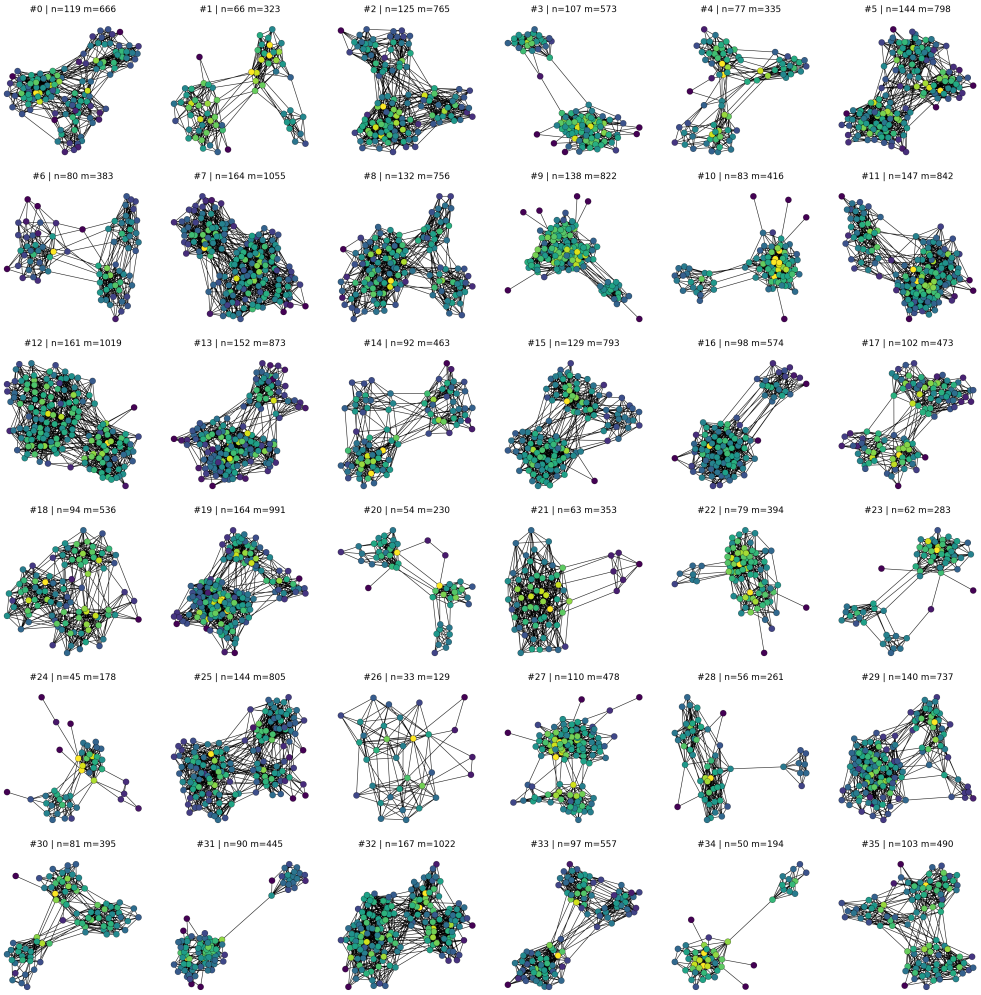

Figure 8: Graphs generated by Flowette for the SBM synthetic datasets.

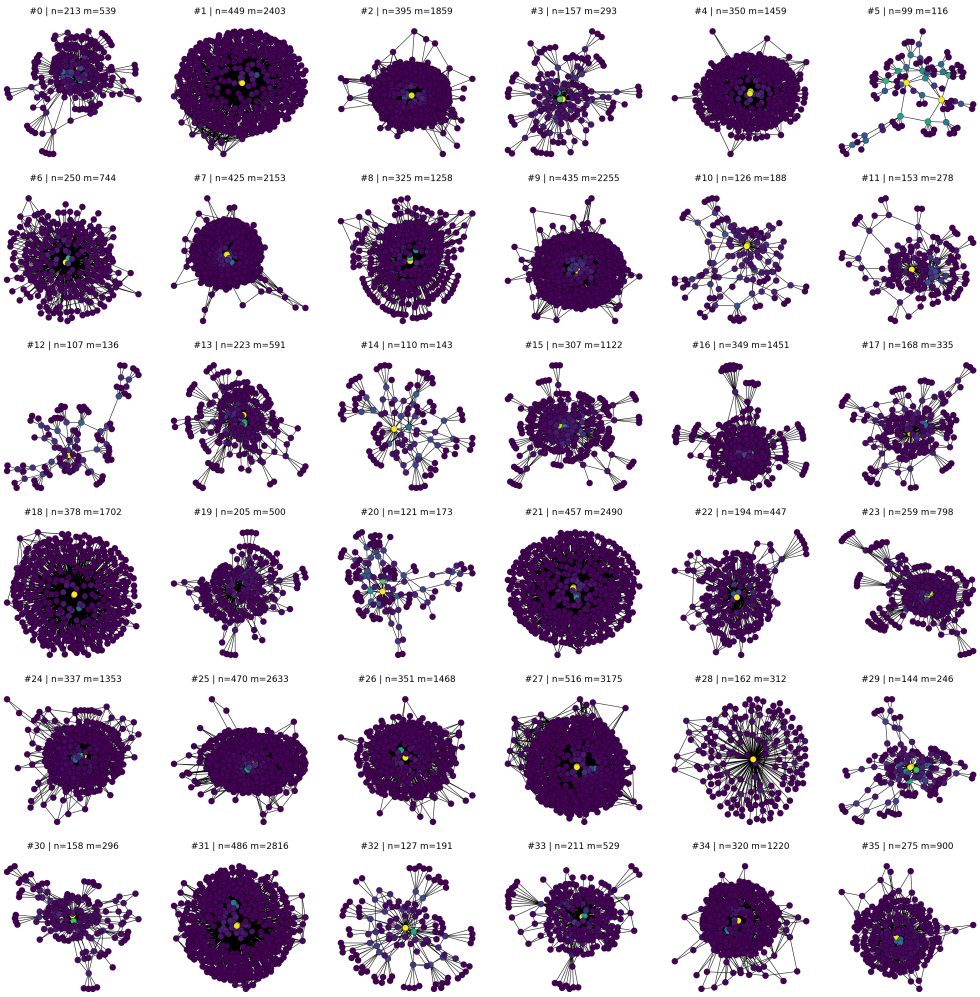

Figure 9: Graphs generated by Flowette for the Ego-small synthetic datasets.

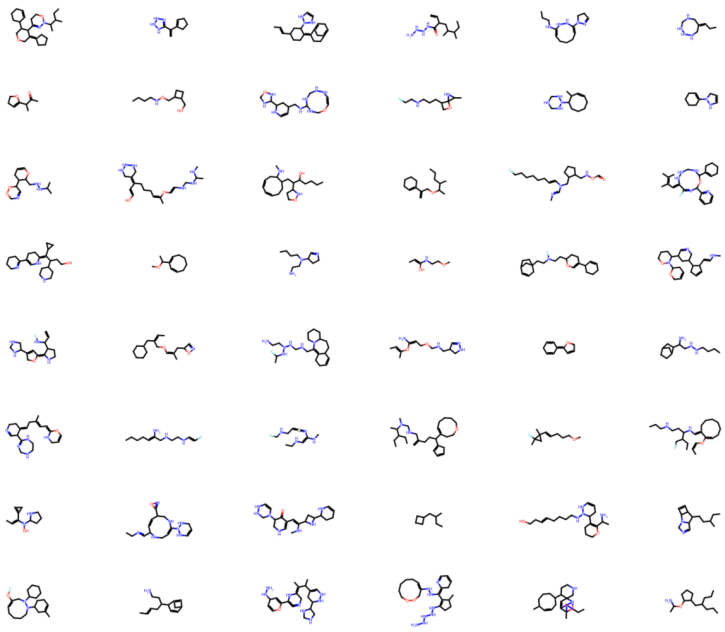

Figure 10: Graphs generated by Flowette for the ZINC250k molecular datasets.

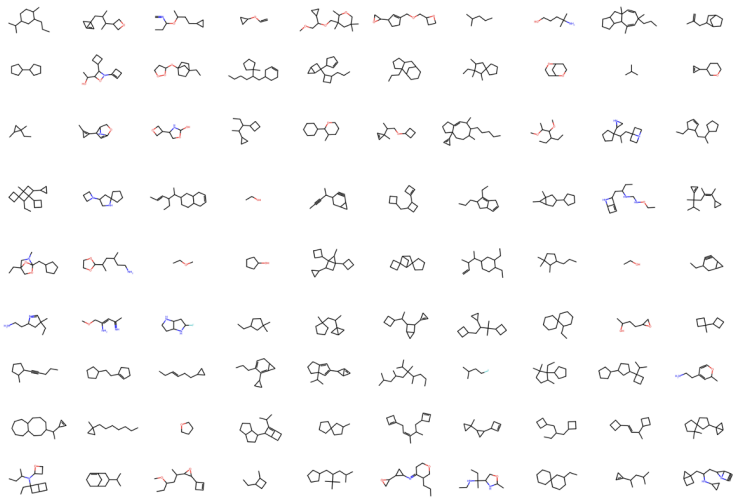

Figure 11: Graphs generated by Flowette for the QM9 molecular datasets.

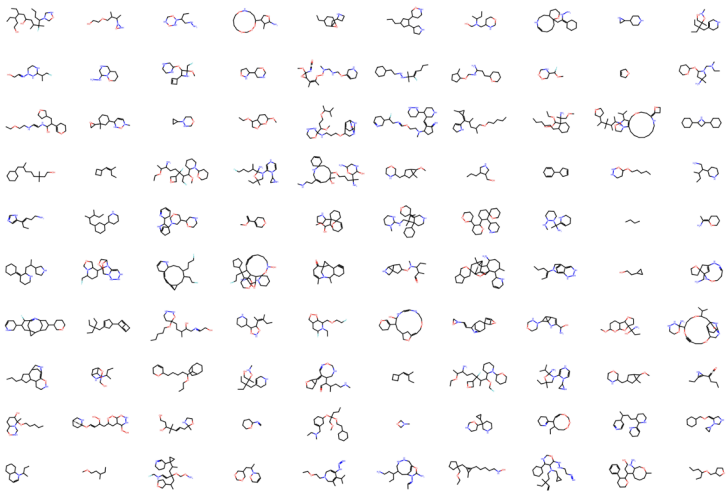

Figure 12: Graphs generated by Flowette for the Guacamol molecular datasets.

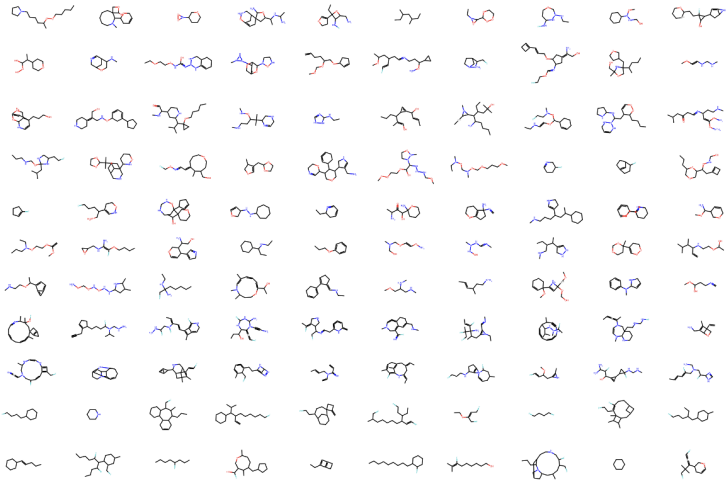

Figure 13: Graphs generated by Flowette for the MOSES molecular datasets.

NeurIPS Paper Checklist

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

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

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

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

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

IMPORTANT, please:

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

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction accurately convey the paper's scope and contributions, clearly outlining the key claims, underlying assumptions, and limitations.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

Justification: The limitations are discussed under "Observed Limitations" on page 10.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

   Justification: A complete set of assumptions, theorems, and proofs is provided in Sections 3 and 4, as well as Appendices E and H.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The paper provides all the information needed to reproduce the main experimental results on pages 8-9, and in Appendix J.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: The paper provides open access to the data and code on page 8. The code is available at: `https://github.com/wokas36/Flowette`

   Guidelines:

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

- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Yes, all the training and test details are provided on pages 8-10, and Appendix J.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: Mean and standard-deviation is provided in many experimental results.

   Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The paper provides sufficient information on computational resources and runtime characteristics to support reproducibility.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: Yes, this work adheres to the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: Not required for our work.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

Justification: The paper does not present any such risks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All datasets and models are properly cited and are publicly available.

    Guidelines:

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

13. **New assets**

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

    Answer: [N/A]

    Justification: No new assets introduced.

    Guidelines:

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

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

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

Answer: [N/A]

Justification: Not used in our work.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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

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

Answer: [N/A]

Justification: Not required for our work.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: Doesn't involve LLMs.

Guidelines:

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