Bures-Wasserstein Flow Matching for Graph Generation

Keyue Jiang* **Aaron Cui** **Laura Toni**
University College London, London, UK

Xiaowen Dong
University of Oxford, Oxford, UK

Abstract

Graph generation has emerged as a critical task in fields ranging from molecule design to drug discovery. Contemporary approaches, notably diffusion and flow-based models, have achieved solid graph generative performance through constructing a probability path that interpolates between a reference distribution and the data distribution. However, these methods typically model the evolution of individual nodes and edges independently and use linear interpolations to build the path assuming that the data lie in Euclidean space. We show that this is suboptimal given the intrinsic non-Euclidean structure and interconnected patterns of graphs, and it poses risks to the sampling convergence. To build a better probability path, we model the joint evolution of the nodes and edges by representing graphs as connected systems parameterized by Markov random fields (MRF). We then leverage the optimal transport displacement between MRF objects to design the probability path for graph generation. Based on this, we introduce BWFlow, a flow-matching framework for graph generation that respects the underlying geometry of graphs and provides smooth velocities in the probability path. The novel framework can be adapted to both continuous and discrete flow-matching algorithms. Experimental evaluations in plain graph generation and 2D/3D molecule generation validate the effectiveness of BWFlow in graph generation with competitive performance, stable training, and guaranteed sampling convergence.

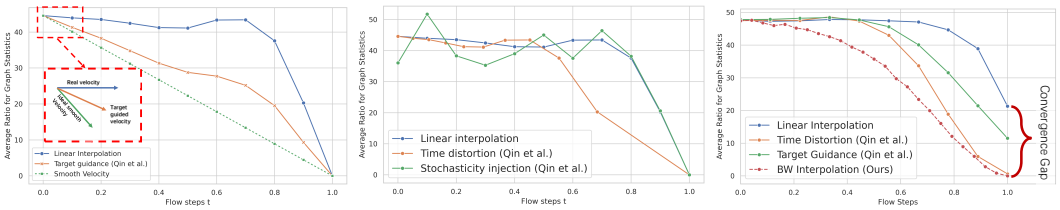
1 Introduction

Thanks to the capability of graphs in representing complex relationships, graph generation [64, 35] has become an essential task in various fields such as protein design [26], drug discovery [6], and social network analysis [31]. Among contemporary generative models, diffusion and flow models have emerged as two compelling approaches for their ability to achieve state-of-the-art performance in graph generation [40, 54, 18, 44, 25]. In particular, these generative models can be unified under the framework of stochastic interpolation [1], which consists of four procedures [34]: 1) Drawing samples from the reference (source) distribution $p_0(\cdot)$ and/or the data (target) distribution $p_1(\cdot)$ for training set assembly; 2) Constructing a time-continuous probability path $p_t(\cdot), 0 \leq t \leq 1$ interpolating between p_0 and p_1; 3) Training a model to reconstruct the probability path by either approximating the score function or velocity fields (ratio matrix in the discrete case); and 4) sampling from p_0 and transforming it through the learned probability path to get samples that approximately follow p_1.

A core challenge in this framework is constructing the probability path p_t. Existing text and image generative models, operating either in the continuous [24, 48, 33, 36] or discrete [8, 50, 9, 19, 39] space, typically rely on linear interpolation between source and target distributions to construct the path. Graph generation models, including diffusion [40, 54, 23, 59, 47] and flow-based models [18, 44, 25], inherit this design by modeling every single node and edge independently and linearly build paths in the disjoint space. However, this approach is inefficient because it neglects the strong

*keyue.jiang.18@ucl.ac.uk

Preprint. Under review.



(a) Training path comparison (b) Training path manipulation (c) Sampling path comparision

Figure 1: Probability path visualization. Since the probability is intractable, the average maximum mean discrepancy ratio (y-axis) of graph statistics between interpolants and the data points is used as a proxy for the probability. Lower means closer to the data distribution (details in Appendix I.3).

interactions and relational structure inherent in graphs, i.e., the significance of a node heavily depends on the configuration of its neighbors. While empirical success have been achieved via fine-grained searching on the training and sampling design [44] such as target guidance and time distortion, we argue that there remains a fundamental issue of the linear probability path construction, and these strategies only mitigate the problem by manipulating the probability path.

Motivation examples. The blue line in Fig. 1a illustrates the probability path evolution through linear interpolation in plain graph generation, where the probability path remains flat until $t \approx 0.8$ before sharply dropping. This pattern provides a poor velocity[2] estimation, as ideally the velocity field should be smooth and consistently pointing to the data distribution as the green line in Fig. 1a [29]. The resultant velocity will make the sampling difficult to converge to the data distribution, as shown in Fig. 1c, where the termination point of the blue curve remains high. Though not explicitly mentioned, the superior performance achieved by Qin et al. [44] is partially attributed to an extensive design space search to manipulate the path for smoother velocity estimation. The techniques they used, including target guidance, time distortion, and stochasticity injection, are conceptually visualized in Fig. 1a and 1b with discussions in Appendix F.1. More motivating examples in Appendix I.2.

The limitations. The above examples reveal fundamental issues of the probability path construction in graph generation, attributable to two primary reasons: 1) The assumption of independence between nodes/edges and a linear interpolation in the locally disjoint space fails to capture the global co-evolution of graph components and properties such as community structure or spectrum [22]. This potentially causes a sudden transition from reference to data distribution, which yields non-smooth probability paths. 2) The linear interpolation is derived through the optimal transport (OT) displacement [53] between distributions residing in the Euclidean space [34]. However, linear interpolation would stray away from the true data manifold when the underlying space is non-Euclidean [11, 29]. Since graphs naturally inhabit non-Euclidean geometries, linearly interpolating the nodes/edges neither guarantees an OT displacement nor respects the underlying geometry, making the constructed probability path suboptimal or even deviate from the valid graph domain.

Proposed solution. To address these limitations, we draw on statistical relational learning and model graphs using Markov Random Fields (MRFs) [52, 45]. MRFs organize the nodes/edges as an interconnected system and interpolating between two MRFs captures the joint evolution of the whole graph system. Extending [22], we derive a closed-form Wasserstein distance between graph distributions and leverage it to construct the Bures-Wasserstein (BW) interpolation of two graphs that ensures the OT displacement in graph generation compared to linear interpolation. We then integrate these insights into a flow-matching framework[3] called Bures–Wasserstein Flow (BWFlow). Specifically, by defining a probability path via BW interpolation, we obtain smooth, globally coherent velocity fields at intermediate steps (see Fig. 1c) that respect the non-Euclidean, interconnected structure of graphs. Crucially, BWFlow admits simulation-free computation of densities and velocities along the entire path, which translates into efficient, stable training and sampling.

Contributions. **First**, we theoretically and empirically show that the linear interpolation used in existing graph generation models gives suboptimal probability path construction and velocity estimation. **Second**, through parameterizing graphs as MRFs, we introduce BWFlow, a flow-matching model for graph generation that constructs probability paths respecting the graph geometry and develops smooth velocities. **Third**, BWFlow was tested on plain graph and 2D/3D molecule

[2]Informally, velocity can be viewed as the path tangent. Section 2 gives a more accurate definition.

[3]For conciseness, we focus on flow models, but the idea is generalizable to diffusions as in Appendix F.2.

generation, demonstrating competitive performance without an excessive search for path manipulation techniques. We further show that BW interpolation consistently outperforms other interpolation methods in building flow matching models, leading to more stable training and sampling convergence.

2 Preliminaries

2.1 Flow matching for graph generation

Flow matching (FM). Generative modeling considers fitting a mapping from state space $\mathcal{S} \to \mathcal{S}$ that transforms the samples from source distribution, $X_0 \sim p_0$, to samples from target data distribution, $X_1 \sim p_1$[4]. Continuous normalizing flow (CNF) [12] parameterizes the transformation through a push-forward equation that interpolates between p_0 and p_1 and constructs a probability path $p_t(\mathcal{X}) = [\psi_t p_0](\mathcal{X})$ through a time-dependent function ψ_t (a.k.a flow). A vector field u_t, defined as $\frac{d}{dt}\psi_t(\mathcal{X}) = u_t(\psi_t(\mathcal{X}))$ with $\psi_0(\mathcal{X}) = \mathcal{X}$, is said to generate p_t if ψ_t satisfies $X_t := \psi_t(X_0) \sim p_t$ for $X_0 \sim p_0$. The FM [33] is designed to match the real velocity field through the loss:

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t, X_t \sim p_t(\cdot)} \left\| v_\theta(X_t) - u_t(X_t) \right\|^2. \tag{1}$$

where $v_\theta(\cdot) : \mathcal{S} \to \mathcal{S}$ is the parameterized velocity field and $t \sim \mathcal{U}[0, 1]$.

Conditional flow matching (CFM). Given that the actual velocity field and the path are not tractable [53], one can construct the per-sample conditional flow. We condition the probability paths on variable $Z \sim \pi(\cdot)$ (for instance, a pair of source and target points $Z = (X_0, X_1)$) and re-write $p_t(\mathcal{X}) = \mathbb{E}_{\pi(\cdot)} p_t(\mathcal{X} \mid Z)$ and $u_t(\mathcal{X}) = \mathbb{E}_{\pi(\cdot)} u_t(\mathcal{X} \mid Z)$ where the conditional path and the velocity field are tractable. The CFM aims at regressing a velocity $v_\theta(\cdot)$ to $u_t(\mathcal{X} \mid Z)$ by the loss,

$$\mathcal{L}_{\text{CFM}}(\theta) := \mathbb{E}_{t, Z \sim \pi(\cdot), p_t(\cdot \mid Z)} \left\| v_\theta(X_t) - u_t(X_t \mid Z) \right\|^2, \tag{2}$$

where it is shown that the CFM optimization has the same optimum as the FM objective [53].

Graphs as statistical objects. When considering graph generation with CFM, the very first step is to model graphs as statistical objects. For notation, we let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{X}\}$ denote an undirected graph random variable with edges $\mathcal{E} = \{e_{uv}\}$, nodes $\mathcal{V} = \{v\}$, and node features $\mathcal{X} = \{x_v\}$. A graph realization is denoted as $G = \{V, E, X\} \sim p(\mathcal{G})$. We consider a group of latent variables that controls the graph distribution, specifically the node feature mean $\boldsymbol{X} = \left[\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{|\mathcal{V}|}\right]^\top \in \mathbb{R}^{|\mathcal{V}| \times K}$, the weighted adjacency matrix $\boldsymbol{W} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$, and the Laplacian matrix $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$, with $\boldsymbol{D} = \text{diag}(\boldsymbol{W}\mathbf{1})$ being the degree matrix (and $\mathbf{1}$ the all-one vector). In a nutshell, graphs are sampled from $G \sim p(\mathcal{G}; \boldsymbol{G}) = p(\mathcal{X}, \mathcal{E}; \boldsymbol{X}, \boldsymbol{W})$.

Graph generation with CFM. The CFM samples new graphs through iteratively building $G_{t+dt} = G_t + v_t^\theta(G_t) \cdot dt$ with initial $G_0 \sim p_0(\mathcal{G})$ and a trained velocity field $v_t^\theta(G_t)$, so that the medium points follows $G_t \sim p_t(\mathcal{G})$ and terminates at $p_1(\mathcal{G})$. We can parameterize $v_t^\theta(G_t)$ as in [19],

$$v_t^\theta(G_t) = \mathbb{E}_{G_0 \sim p_0(\mathcal{G}), G_1 \sim p_{1|t}^\theta(\cdot \mid G_t)} \left[v_t(G_t \mid G_0, G_1) \right] \tag{3}$$

As such, training the velocity fields is replaced by training a denoiser $p_{1|t}^\theta(\cdot \mid G_t)$ to predict the clean datapoint, which is equivalent to maximizing the log-likelihood [44, 9],

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{G_1 \sim p_1(\cdot), G_0 \sim p_0(\cdot), t \sim \mathcal{U}_{[0,1]}(\cdot), G_t \sim p_{t|0,1}(\cdot \mid G_1, G_0)} \left[\log p_{1|t}^\theta(G_1 \mid G_t) \right] \tag{4}$$

where t is sampled from a uniform distribution $\mathcal{U}_{[\cdot]}$ and $G_t \sim p_{t|0,1}$ can be obtained in a simulation-free manner. This framework avoids the evaluation of the conditional vector field at training time, which both increases the model robustness and training efficiency.

To proceed, a closed form of $p_t(\cdot \mid G_0, G_1)$ is required to construct both the probability path and the velocity field $v_t(G_t \mid G_0, G_1)$. A common selection to decompose the probability density assumes independency for each node and edge [25, 44, 18] giving $p(\mathcal{G}) = p(\mathcal{X})p(\mathcal{E}) = \prod_{v \in \mathcal{V}} p(x_v) \prod_{e_{uv} \in \mathcal{E}} p(e_{uv})$.

[4]For clarity, we denote the calligraphic style $\mathcal{X}$ being the random variable, the plain X the relevant realizations and the bold symbol $\boldsymbol{X}$ the latent variables (parameters) that controls the distributons, i.e. $X \sim p(\mathcal{X}; \boldsymbol{X})$.

Choosing $\pi(\cdot) = p_0(\mathcal{G})\,p_1(\mathcal{G})$, the boundary conditions follow $p_i(\mathcal{G}) = \delta(\mathcal{X}_i = \boldsymbol{X}_i)\cdot\delta(\mathcal{E}_i = \boldsymbol{W}_i), \forall i = \{0,1\}$ with δ the dirac function. This decomposition is further combined with linear interpolation to build the path, as introduced in [53], where,

$$\begin{aligned} p_t(x_v \mid G_0, G_1) &= \mathcal{N}\left(t[X_1]_v + (1-t)[X_0]_v, \sigma_t^2\right) \text{ and } u_t(x_v \mid X_0, X_1) = [X_1]_v - [X_0]_v, \\ p_t(e_{uv} \mid G_0, G_1) &= \mathcal{N}\left(t[E_1]_{uv} + (1-t)[E_0]_{uv}, \sigma_t^2\right) \text{ and } u_t(e_{uv} \mid G_0, G_1) = [E_1]_{uv} - [E_0]_{uv}. \end{aligned} \tag{5}$$

Similarly, discrete flow matching frameworks for graph generation [44, 47, 59] is also based on linear interpolation, where the interpolant is sampled from a categorical distribution whose probabilities are simply linear interpolation between the boundary conditions.

2.2 Optimal transport and flow matching

Why linear interpolation? Existing literature [36, 1] argues that the probability path $p_t(\mathcal{X} \mid Z)$ should be chosen to recover the optimal transport (OT) displacement interpolant [38]. The (Kantorovich) optimal transport problem is to find the transport plan between two probability measures, η_0 and η_1, with the smallest associated transportation cost defined as follows.

Definition 1 (Wasserstein Distance)**.** Denote the possible coupling as $\pi \in \Pi(\eta_0, \eta_1)$, which is a joint measure on $\mathcal{S}\times\mathcal{S}$ whose marginals are η_0 and η_1 respectively. With $c(X,Y)$ being the cost of transporting the mass between X and Y, the Wasserstein distance is defined as,

$$\mathcal{W}_c(\eta_0,\eta_1) = \inf_{\pi\in\Pi(\eta_0,\eta_1)} \int_{\mathcal{S}\times\mathcal{S}} c(X,Y)\,d\pi(X,Y). \tag{6}$$

When the data follow Euclidean geometry and both boundary distributions, p_0 and p_1 are described by the Gaussian family, the probability path shown in Eq. (5) with $\sigma_t \to 0$ becomes a solution to Eq. (6).

As suggested in the motivations, linearly interpolating in the disjoint space of nodes and edges with Eq. (5) does not guarantee the OT displacement in non-Euclidean and interconnected objects like graphs. To overcome the limitation, we utilize Markov random fields to capture the joint evolution of the graph system, and build an FM model that generates graphs with smooth probability paths and consistent velocity.

3 Methodology

In this paper, we introduce Bures–Wasserstein Flow Matching (BWFlow), a novel graph generation framework that is built upon the OT displacement when modeling graphs with Markov Random Fields (MRFs). We begin by casting graphs in an MRF formulation in Section 3.1. We then derive the BWFlow framework in Section 3.2 by formulating and solving the OT displacement problem on the MRF, thereby yielding the fundamental components, interpolations and velocity fields, for FM-based graph generation. Finally, in Section 3.3, we extend BWFlow to discrete FM regimes, enabling its application across a broad spectrum of graph-generation tasks. A schematic overview of the entire BWFlow is illustrated in Fig. 2.

3.1 Graph Markov random fields

We borrow the idea from MRF as a remedy to modeling the complex system organized by graphs, which intrinsically captures the underlying mechanism that jointly generates the nodes and edges. Mathematically, we assume the joint probability density distribution (PDF) of node features and graph structure as $p(\mathcal{G};\boldsymbol{G}) = p(\mathcal{X},\mathcal{E};\boldsymbol{X},\boldsymbol{W}) = p(\mathcal{X};\boldsymbol{X},\boldsymbol{W})p(\mathcal{E};\boldsymbol{W})$ where the node features and graph structure are interconnected through latent variables $\boldsymbol{X}$ and $\boldsymbol{W}$. For node features $\mathcal{X}$, we follow the MRF assumption in Zhu et al. [63] and decompose the density into the node-wise potential $\varphi_1(v), \forall v \in \mathcal{V}$ and pair-wise potential $\varphi_2(u,v), \forall e_{uv} \in \mathcal{E}$:

$$p(\mathcal{X};\boldsymbol{X},\boldsymbol{W}) \propto \prod_v \underbrace{\exp\left\{-(\nu+d_v)\|\boldsymbol{V}x_v - \boldsymbol{\mu}_v\|^2\right\}}_{\varphi_1(v)} \prod_{u,v} \underbrace{\exp\left\{w_{uv}\left[(\boldsymbol{V}x_u - \boldsymbol{\mu}_u)^\top(\boldsymbol{V}x_v - \boldsymbol{\mu}_v)\right]\right\}}_{\varphi_2(u,v)}, \tag{7}$$

with $\|\cdot\|$ the L_2 norm, $(\cdot)^\dagger$ the pseudo-inverse, $\boldsymbol{V}$ the transformation matrix modulating the graph feature emission, and $\boldsymbol{\mu}_v$ the node-specific latent variable mean. Eq. (7) can be expressed as a colored

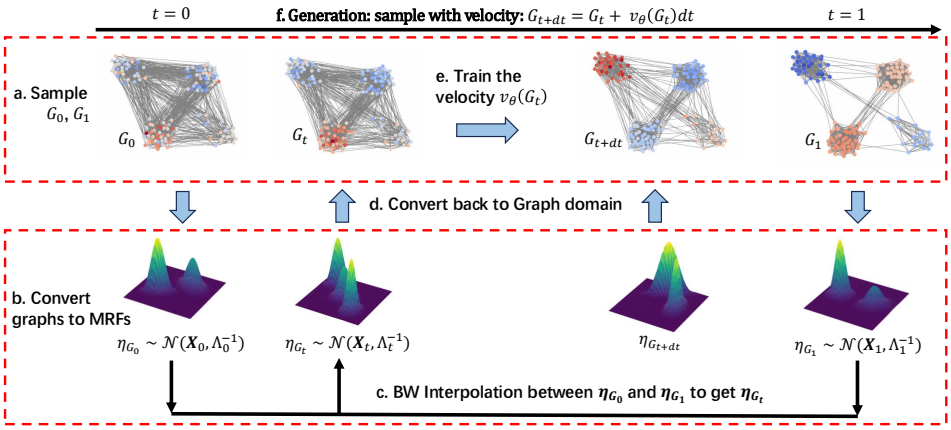


Figure 2: Schematic overview of BWFlow, which consists of: a) Sample the marginal graph condition G_0 and G_1; b) Convert graphs to MRFs; c) Interpolate to get intermediate points; d) Convert back to get G_t; e) Train velocity based on G_t; and f) Generate new points with the trained velocity.

Gaussian distribution in Eq. (8) given that $\boldsymbol{V}x_v \sim \mathcal{N}(\boldsymbol{\mu}_v, (\nu\boldsymbol{I} + \boldsymbol{L})^{-1})$. We further assume that edges are emitted via a Dirac delta, $\mathcal{E} \sim \delta(\boldsymbol{W})$, yielding our definition of Graph Markov Random Fields (GraphMRF). The derivation can be found in Appendix A.2.

Definition 2 (Graph Markov Random Fields)**.** GraphMRF statistically describes graphs as,

$$p(\mathcal{G}; \boldsymbol{G}) = p(\mathcal{X}, \mathcal{E}; \boldsymbol{X}, \boldsymbol{W}) = p(\mathcal{X}; \boldsymbol{X}, \boldsymbol{W}) \cdot p(\mathcal{E}; \boldsymbol{W}) \text{ where } \mathcal{E} \sim \delta(\boldsymbol{W}) \text{ and}$$
$$\operatorname{vec}(\mathcal{X}) \sim \mathcal{N}\left(\boldsymbol{X}, \boldsymbol{\Lambda}^{\dagger}\right), \text{ with } \boldsymbol{X} = \operatorname{vec}(\boldsymbol{V}^{\dagger}\boldsymbol{\mu}), \boldsymbol{\Lambda} = (\nu\boldsymbol{I} + \boldsymbol{L}) \otimes \boldsymbol{V}^{\top}\boldsymbol{V}. \tag{8}$$

The $\otimes$ is the Kronecker product, $\operatorname{vec}(\cdot)$ is the vectorization operator and $\boldsymbol{I}$ is the identity matrix.

Remark 1. GraphMRF explicitly captures node–edge dependencies and preserves the advantages of colored Gaussian distributions. Section 3.2 will soon show that this yields closed-form interpolation and velocity, and the probability path constructed from GraphMRFs remains on the graph manifold that respects the underlying non-Euclidean geometry.
Remark 2. While we emphasize that GraphMRF is not a universal model and imposes certain constraints (Appendix A.3 discusses about the usage scope), it nonetheless captures the dynamics of most graph-generation tasks such as planar, stochastic block models, and molecular graphs.

3.2 Bures-Wasserstein flow matching for graph generation

The optimal transport displacement between graph distributions. Given that the joint probability of graphs decomposed as $p(\mathcal{G}) = p(\mathcal{X}; \boldsymbol{X}, \boldsymbol{W})p(\mathcal{E}; \boldsymbol{W})$ and the measure factorized to $\eta_{\mathcal{G}_j} = \eta_{\mathcal{X}_j} \cdot \eta_{\mathcal{E}_j}$ with $j \in \{0, 1\}$, the graph Wasserstein distance between $\eta_{\mathcal{G}_0}$ and $\eta_{\mathcal{G}_1}$ is written as,

$$\text{(Graph Wasserstein Distance)} \quad d_{\mathrm{BW}}(\mathcal{G}_0, \mathcal{G}_1) := \mathcal{W}_c\left(\eta_{\mathcal{G}_0}, \eta_{\mathcal{G}_1}\right) = \mathcal{W}_c(\eta_{\mathcal{X}_0}, \eta_{\mathcal{X}_1}) + \mathcal{W}_c(\eta_{\mathcal{E}_0}, \eta_{\mathcal{E}_1}).$$

We extend Haasler and Frossard [22] and analytically derive the graph Wasserstein distance using the OT formula between Gaussians [16, 41, 51] (see Lemma 2 proved in Appendix B.1) as follows.

Proposition 1 (Bures-Wasserstein Distance)**.** *Consider two same-sized graphs $\mathcal{G}_0 \sim p(\mathcal{X}_0, \mathcal{E}_0)$ and $\mathcal{G}_1 \sim p(\mathcal{X}_1, \mathcal{E}_1)$ with $\boldsymbol{V}$ shared for two graphs, described by the distribution in Definition 2. When the graphs are equipped with graph Laplacian matrices $\boldsymbol{L}_0$ and $\boldsymbol{L}_1$ satisfying 1) is Positive Semi-Definite (PSD) and 2) has only one zero eigenvalue. The Bures-Wasserstein distance between these two random graph distributions is given by*

$$d_{BW}(\mathcal{G}_0, \mathcal{G}_1) = \|\boldsymbol{X}_0 - \boldsymbol{X}_1\|_F^2 + \beta \operatorname{trace}\left(\boldsymbol{L}_0^{\dagger} + \boldsymbol{L}_1^{\dagger} - 2\left(\boldsymbol{L}_0^{\dagger/2}\boldsymbol{L}_1^{\dagger}\boldsymbol{L}_0^{\dagger/2}\right)^{1/2}\right), \tag{9}$$

as $\nu \to 0$ and β is a constant related to the norm of $\boldsymbol{V}^{\dagger}$. The proof can be found in Appendix B.2.

Based on the Bures-Wasserstein (BW) distance, we then derive the OT interpolant for two graphs, which is the solution of the displacement minimization problem described as,

$$\mathcal{G}_t = \arg\min_{\tilde{\mathcal{G}}} (1-t) d_{\mathrm{BW}}(\mathcal{G}_0, \tilde{\mathcal{G}}) + t d_{\mathrm{BW}}(\tilde{\mathcal{G}}, \mathcal{G}_1). \tag{10}$$

The probability path. The interpolation is obtained through solving Eq. (10) with the BW distance defined in Proposition 1, we prove the minimizer of the above problem has the form in Proposition 2. The proof can be found in Appendix C.1.

Proposition 2 (Bures-Wasserstein interpolation)**.** *The graph minimizer of Eq.* (10)*, $\mathcal{G}_t = \{\mathcal{V}, \mathcal{E}_t, \mathcal{X}_t\}$, have its node features following a colored Gaussian distribution, $\mathcal{X}_t \sim \mathcal{N}(\boldsymbol{X}_t, \boldsymbol{\Lambda}_t^\dagger)$ with $\boldsymbol{\Lambda}_t = (\nu \boldsymbol{I} + \boldsymbol{L}_t) \otimes \boldsymbol{V}^\top \boldsymbol{V}$ and edges following $\mathcal{E}_t \sim \delta(\boldsymbol{W}_t)$, specifically,*

$$\boldsymbol{L}_t^\dagger = \boldsymbol{L}_0^{1/2} \left((1-t)\boldsymbol{L}_0^\dagger + t \left(\boldsymbol{L}_0^{\dagger/2} \boldsymbol{L}_1^\dagger \boldsymbol{L}_0^{\dagger/2} \right)^{1/2} \right)^2 \boldsymbol{L}_0^{1/2}, \quad \boldsymbol{X}_t = (1-t)\boldsymbol{X}_0 + t\boldsymbol{X}_1 \tag{11}$$

The interpolant provides a closed form for the induced probability path $p(\mathcal{G}_t \mid G_0, G_1)$ and the velocity $v(G_t \mid G_0, G_1)$ that is easy to access without any simulation.

The velocity. We consider the reparameterization as in Eq. (3) and derive the conditional velocity $v_t(G_t \mid G_1, G_0)$ as in Proposition 3.

Proposition 3 (Bures-Wasserstein velocity)**.** *For the graph $\mathcal{G}_t$ following BW interpolation in Proposition 2, the conditional velocity at time t with observation G_t is given as,*

$$\begin{aligned} &v_t(E_t \mid G_0, G_1) = \dot{\boldsymbol{W}}_t = \mathit{diag}(\dot{\boldsymbol{L}}_t) - \dot{\boldsymbol{L}}_t, \quad v_t(X_t \mid G_0, G_1) = \frac{1}{1-t}(\boldsymbol{X}_1 - \boldsymbol{X}_t) \\ &\text{with } \dot{\boldsymbol{L}}_t = 2\boldsymbol{L}_t - \boldsymbol{T}\boldsymbol{L}_t - \boldsymbol{L}_t\boldsymbol{T} \text{ and } \boldsymbol{T} = \boldsymbol{L}_0^{1/2}(\boldsymbol{L}_0^{\dagger/2}\boldsymbol{L}_1^\dagger\boldsymbol{L}_0^{\dagger/2})^{1/2}\boldsymbol{L}_0^{1/2} \end{aligned} \tag{12}$$

where $\boldsymbol{W}_t = \boldsymbol{D}_t - \boldsymbol{L}_t$ and $\boldsymbol{L}_t$ defined in Eq. (11)*. Derivation can be found in Appendix C.2.*

With Proposition 2 and Proposition 3, we are now able to formally construct the algorithms for Bures-Wasserstein flow matching. Taking continuous flow matching as an example, Algorithms 1 and 2 respectively introduce the training and sampling pipelines for our BWFlow.

Remark: The BW interpolation and velocity both deviate from the linear flow matching framework and require extra computational cost. However, there exist multiple ways to analytically calculate or numerically approximate the velocity for training and inference. The choice of these methods depends on the trade-off between training stability, sampling efficiency, etc. In Appendix E, we provide a discussion about the design space of BW interpolation and velocity.

3.3 Discrete Bures-Wasserstein flow matching for graph generation

Up to now we are working on the scenario when $p(\mathcal{X} \mid \boldsymbol{X}, \boldsymbol{W})$ is a Gaussian and $p(\mathcal{E} \mid \boldsymbol{W})$ is a Dirac distribution. However, previous studies have observed a significant improvement of the discrete counterpart of the continuous graph generation models [54, 59, 44]. To benefit our model from such a nature, we derive the discrete Bures-Wasserstein flow matching for graph generation.

The discrete probability path. We design the probability path as discrete distributions,

$$\begin{aligned} &p_t(x_v \mid G_0, G_1) = \mathrm{Categorical}([\boldsymbol{X}_t]_v), \quad p_t(e_{uv} \mid G_0, G_1) = \mathrm{Bernoulli}([\boldsymbol{W}_t]_{uv}) \\ &\text{s.t. } p_0(\mathcal{G}) = \delta(G_0, \cdot), p_1(\mathcal{G}) = \delta(G_1, \cdot) \end{aligned} \tag{13}$$

where $\boldsymbol{W}_t = \boldsymbol{D}_t - \boldsymbol{L}_t$ with $\boldsymbol{X}_t$ and $\boldsymbol{L}_t$ defined the same in Eq. (11). We consider the fact that the Dirac distribution is a special case when the Categorical/Bernoulli distribution has probability 1 or 0, so the boundary condition $p_0(\mathcal{G}) = \delta(G_0, \cdot), p_1(\mathcal{G}) = \delta(G_1, \cdot)$ holds. Even though we are not sampling from Gaussian distributions anymore, it is possible to approximate the Wasserstein distance between two multivariate discrete distributions with the Gaussian counterpart so the conclusions, such as optimal transport displacements, still hold. We left the discussion in Appendix D.2.

The discrete velocity fields. The path of node features $\mathcal{X}_t$ can be re-written as $p_t(\mathcal{X}) = (1-t)\delta(\cdot, \boldsymbol{X}_0) + t\delta(\cdot, \boldsymbol{X}_1)$ so the conditional velocity can be accessed through $v_t(X_t \mid G_0, G_1) =$

Algorithm 1: BWFlow Training

Input: Ref. dist p_0 and dataset $\mathcal{D} \sim p_1$.
Output: Trained model $f_\theta(G_t, t)$.

Initialize model $f_\theta(G_t, t)$;
while f_θ *not converged* **do**
  `/* Sample Boundary Graphs */`
  Sample batched $\{G_0\} \sim p_0$, $\{G_1\} \sim \mathcal{D}$;
  `/* Prob.path Construction */`
  Sample $t \sim \mathcal{U}(0, 1)$;
  Calculate the BW interpolation $p(G_t \mid G_0, G_1)$ via Eq. (11);
  `/* Denoising - x-prediction */`
  $p^\theta_{1|t}(\cdot \mid G_t) \leftarrow f_\theta(G_t, t)$;
  Loss calculation via Eq. (4);
  optimizer.step();

Algorithm 2: BWFlow Sampling

Input: Reference distribution p_0, Trained Model $f_\theta(G_t, t)$, Small time step dt,
Output: Generated Graphs $\{\hat{G}_1\}$.

Initialize samples $\{\hat{G}_0\} \sim p_0$;
Initialize the model $p^\theta_{1|t}(\cdot \mid G_t) \leftarrow f_\theta(G_t, t)$
for $t \leftarrow 0$ **to** $1 - dt$ ***by*** dt **do**
  `/* Denoising - x-prediction */`
  Predict $\tilde{G}_1 \leftarrow p^\theta_{1|t}(\cdot \mid \hat{G}_t)$;
  `/* Velocity calculation */`
  Calculate $v_\theta(\hat{G}_t \mid \hat{G}_0, \tilde{G}_1)$ via Eq. (12);
  `/* Numerical Sampling */`
  Sample $\hat{G}_{t+dt} \sim \hat{G}_t + v_\theta(\hat{G}_t)dt$

$[\delta(\cdot, \boldsymbol{X}_1) - \delta(\cdot, \boldsymbol{X}_t)]/(1-t)$. However, the probability path of edges $\mathcal{E}_t$, shown in Eqs. (11) and (13), cannot be written as a mixture of two boundary conditions given the non-linear interpolation. To this end, we derive in Appendix D.3 that the discrete velocity follows,

$$v_t(E_t \mid G_1, G_0) = (1 - 2E_t)\, \frac{\dot{\boldsymbol{W}}_t}{\boldsymbol{W}_t \circ (1 - \boldsymbol{W}_t)}, \tag{14}$$

where $\boldsymbol{W}_t = \boldsymbol{D}_t - \boldsymbol{L}_t$, $\dot{\boldsymbol{W}}_t = \mathrm{diag}(\dot{\boldsymbol{L}}_t) - \dot{\boldsymbol{L}}_t$ with $\boldsymbol{L}_t$, $\dot{\boldsymbol{L}}_t$ defined in Eqs. (11) and (12) respectively. With the interpolation and velocity defined, the discrete flow matching is built in Algorithms 3 and 4.

4 Experiments

We evaluate the BWFlow algorithms through both the plain graph generation and real-world molecule generation tasks. We first outline the experimental setup in Section 4.1, followed by a general comparison in Section 4.2. Next, we conduct behavior analysis on the impact of interpolation methods and the corresponding velocity construction on graph generation performance in Section 4.3, which demonstrates the effectiveness and benefit of flow along Bures-Wasserstein interpolation.

4.1 Experiment settings

Dataset. For plain graph generation, we evaluate the quality of generated graphs on three benchmark datasets following previous works [37, 54, 4], specifically, **planar** graphs, **tree** graphs, and stochastic blocking models (**SBM**). Two datasets, namely MOSES [42] and GUACAMOL [7], are benchmarked to test the model performance on 2D molecule generation. For 3D molecule generation with coordinate data, we test the model on QM9 [46] and GEOM-DRUGS [2].

Metrics. In plain graph generation, the evaluation metrics include the percentage of Valid, Unique, and Novel (**V.U.N.**) graphs, and the average maximum mean discrepancy ratio (**A.Ratio**) of graph statistics between the set of generated graphs and the test set are reported (details in Appendix I.3). For molecule generation, we test two scenarios with and without bond type information, where the latter validates the capacity of our methods in generating the graph structures. To this end, we develop a new relaxed metric to measure the stability and validity of atoms and molecules when bond types are not available. Specifically, the atom-wise stability is relaxed as:

$$\text{Stability of Atom } i\text{: } s_i = \mathbb{I}[\exists \{b_{ij}\}_{j \in \mathcal{N}_i} \in \prod_{j \in \mathcal{N}_i} B_{ij} : \sum_{j \in \mathcal{N}_i} b_{ij} = \mathrm{EV}_i], \text{ with the identity function } \mathbb{I}.$$

This means atom i is "relaxed-stable" if there is at least one way to pick allowed bond types (B_{ij}) to its neighbors $\mathcal{N}_i$ so that their total exactly matches the expected valences EV_i. Such a relaxed stability of atoms (**Atom.Stab.**) inherently defines molecule stability (**Mol.Stab.**) and the **validity** of a molecule, which are the shared metrics for both 2D/3D molecule generation. In addition to these

metrics, distribution metrics are also used for 2D molecules (FCD, Scaf, etc.), and 3D generations (charge distributions, atom total variation, angles, etc.). Details in Appendix I.3.

Setup. To isolate the impact from model architecture, we follow Qin et al. [44] to fix the backbone model as the same graph transformers. It is shown that sampling/training distortion and target guidance have a significant impact on the performance of graph generation tasks [44]. In our experiment, the best model performance is obtained with these technologies, but in behavior analysis, we disabled time distortion and target guidance for a fair comparison. In molecular generation, two scenarios with and without bond types information are considered to better evaluate the ability of generating graph structures. More experimental details can be found in Appendix I.1.

4.2 Main results for graph generation

Plain graph generation. In Table 1, we report both V.U.N. and A.Ratio. As performance on these benchmarks continues to fluctuate significantly even after convergence and the results are near saturated, we present not only the best scores but also the exponentially moving averaged (EMA) results on last 5 checkpoints and decay 0.999. BWFlow outperforms most competitors on Planar and SBM graph generation. The lone exception is the tree graphs, where our model falls short. We attribute this gap to the fundamentally different generation process for tree graphs (which reside in hyperbolic space [60]) and thus violate our MRF assumptions.

Table 1: Plain Graph generation performance. We sampled 5 times (each run generates 40 graphs) to calculate the mean and standard deviation. We only keep the main diffusion/flow model for comparison, while other models are included in the full version at Table 8.

		Planar		Tree		SBM	
Model	Class	V.U.N. ↑	A.Ratio ↓	V.U.N. ↑	A.Ratio ↓	V.U.N. ↑	A.Ratio ↓
Train set	—	100	1.0	100	1.0	85.9	1.0
DiGress [54]	Diffusion	77.5	5.1	90.0	1.6	60.0	1.7
HSpectre [4]	Diffusion	95.0	2.1	**100.0**	4.0	75.0	10.5
GruM [28]	Diffusion	90.0	1.8	—	—	85.0	**1.1**
CatFlow [18]	Flow	80.0	—	—	—	85.0	—
DeFoG [44]	Flow	**99.5**±1.0	1.6±0.4	96.5±2.6	1.6±0.4	90.0±5.1	4.9±1.3
BWFlow	Flow	97.5±2.5	**1.3**±0.4	75.5±2.4	1.4±0.3	**90.5**±4.0	3.8±0.9
DeFoG (EMA)	Flow	77.5±8.37	3.5±1.7	**73.1**±11.4	1.50±0.3	**85.0**±7.1	3.7±0.9
BWFlow (EMA)	Flow	**84.8**±6.44	**2.4**±0.9	68.1±12.2	**1.24**±0.2	83.5±6.0	**2.4**±0.6

2D molecule graph generation. The model performance is illustrated in Table 2. In both datasets, BWFlow can achieve competitive results near the state-of-the-art (SOTA) flow matching models [44], and outperforms the diffusion models. Given that MOSES and GUACAMOL benchmarks are approaching saturation, the fact that BWFlow achieves results on par with the SOTA models serves as strong evidence of its effectiveness.

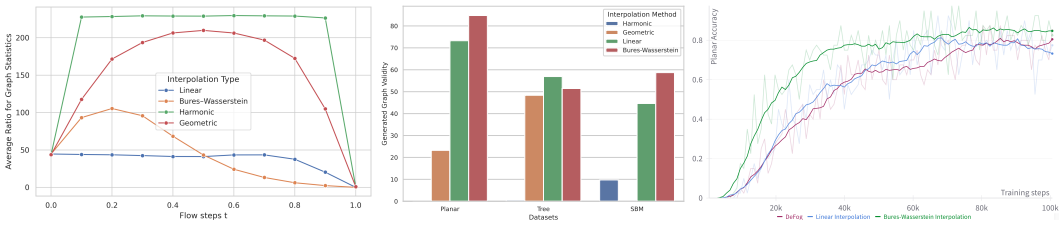
Table 2: Large molecule generation results. Only diffusion and flow models are reported. Table 15 gives further experiments with binary edge types.

	Guacamol				MOSES						
Model	Val. ↑	V.U. ↑	V.U.N.↑	FCD ↑	Val. ↑	Unique. ↑	Novelty ↑	Filters ↑	FCD ↓	SNN ↑	Scaf ↑
Training set	100.0	100.0	0.0	92.8	100.0	100.0	0.0	100.0	0.01	0.64	99.1
DiGress [54]	85.2	85.2	85.1	68.0	85.7	**100.0**	95.0	97.1	**1.19**	0.52	14.8
DisCo [59]	86.6	86.6	86.5	59.7	88.3	**100.0**	**97.7**	95.6	1.44	0.50	15.1
Cometh [47]	98.9	98.9	97.6	72.7	90.5	99.9	92.6	**99.1**	1.27	0.54	**16.0**
DeFoG [44]	**99.0**	**99.0**	**97.9**	**73.8**	**92.8**	99.9	92.1	98.9	1.95	0.55	14.4
BWFlow (Ours)	98.8	98.9	97.4	69.2	92.0	**100.0**	94.5	98.4	1.32	**0.56**	15.3

3D molecule generation. Table 3 gives the results on the 3D molecule generation task with explicit hydrogen, where we ignore the bond type but just view the adjacency matrix as a binary one for validating the power of generating graph structures. Interestingly, the empirical results show that even without edge type, the 3D graph generation model already can capture the molecule data distribution. And our BWFlow significantly outperforms the SOTA models, including MiDi [55] and FlowMol [17]. We believe a promising future direction is to incorporate the processing of multiple bond types into our framework, which would potentially raise the performance by a margin.

Dataset	Interpolation	Metrics							
		μ	V.U.N(%)	Mol.Stab.	Atom.Stab.	Connected(%)	Charge(10^{-2})	Atom(10^{-2})	Angles(°)
QM9 (with h)	MiDi	1.01	93.13	93.98	99.60	99.21	0.2	3.7	2.21
	FlowMol	1.01	87.53	88.45	99.13	99.09	0.4	4.2	2.72
	BWFlow	1.01	**96.45**	**97.84**	**99.84**	**99.24**	**0.1**	**2.3**	**1.96**
GEOM (with h)	Midi	1.34	78.23	32.42	89.61	**79.15**	0.6	11.2	9.6
	FlowMol	1.34	82.20	36.90	94.60	59.98	0.4	8.8	6.5
	BWFlow	**1.20**	**87.75**	**46.80**	**95.08**	73.53	**0.1**	**6.5**	**3.96**

Table 3: Quantitative experimental results on 3D Molecule Generation with explicit hydrogen.



(a) The evolution of graph statistics ratio along the probability path.

(b) The impact of interpolation methods on the performance.

(c) Convergence analysis of BW-Flow and flows with linear interpolations.

Figure 3: Ablation studies for Bures-Wasserstein Flow Matching.

4.3 Behavior analysis

BWFlow provides smooth velocity in probability paths. To illustrate how BWFlow models the smooth evolution of graphs, we compute the A.Ratio on SBM datasets (the figures for the others are in Fig. 5) between generated graph interpolants and test data for $t \in [0, 1]$, as shown in Fig. 3a. In contrast to the linear (arithmetic) interpolation, BW interpolation initially exposes the model to more out-of-distribution samples with increased A.Ratio. After this early exploration, the A.Ratio monotonously converges, yielding a smooth interpolation between the reference graphs and the data points. This behavior enhances both the model robustness and velocity estimation, which helps in covering the convergence gap in the generation stage as in Fig. 1c. In comparison, harmonic and geometric interpolations step outside the valid graph domain, making the learning ill-posed.

The impact of interpolation metrics on the model performance. Fig. 3b illustrated a bar plot that compares interpolation methods on the ability of generating valid plain graphs measured by V.U.N., which shows the superiority of BW interpolation in capturing graph distributions. Fig. 3c illustrated an example (in planar graph generation) of the convergence curve at the training stage (full results in Table 10), which suggests that BWFlow can bring a faster convergence speed compared to FM methods constructed with linear (arithmetic) interpolations.

5 Discussion and future work

In this paper, we introduce BWFlow, a flow matching model that captures the non-Euclidean and interconnected properties of graphs. While we show BWFlow exhibits outstanding performance in various graph generation tasks, it faces the following limitations that motivate solid future work.

Extension to multiple relation types. As our framework is built upon the interpolation parameterized by the Graph Laplacian, it is not easily generalizable to the graph generation with multiple edge types. We made preliminary attempts at the extension but a comprehensive design is still required.

Lower computational complexity. While constructing the probability path and the velocity, our BW interpolation suffers from an extra cost due to its request to compute the pseudo-inverse of the Laplacian. Compared to linear interpolation, this brings $O(N^3)$ extra complexity theoretically, and empirically 2x training time and inference time, which is non-negligible in large graph generation. We aim to develop iterative optimization methods to make training faster in future work.

More universal interpolation that accommodates the geometry. In our experiments on the tree dataset, performance was unsatisfactory. We attribute this issue to the unique geometry of tree-structured graphs. A promising future work includes selecting the adaptive interpolation schemes that accommodate the intrinsic geometry of a graph.

References

[1] Michael S. Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *ICLR*. OpenReview.net, 2023.

[2] Simon Axelrod and Rafael Gómez-Bombarelli. GEOM: energy-annotated molecular conformations for property prediction and molecular generation. *CoRR*, abs/2006.05531, 2020.

[3] Eric Bach, Simon Rogers, John Williamson, and Juho Rousu. Probabilistic framework for integration of mass spectrum and retention time information in small molecule identification. *Bioinformatics*, 37(12):1724–1731, 11 2020. ISSN 1367-4803. doi: 10.1093/bioinformatics/btaa998. URL https://doi.org/10.1093/bioinformatics/btaa998.

[4] Andreas Bergmeister, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Efficient and scalable graph generation through iterative local expansion. In *ICLR*. OpenReview.net, 2024.

[5] Rajendra Bhatia, Tanvi Jain, and Yongdo Lim. On the bures–wasserstein distance between positive definite matrices. *Expositiones Mathematicae*, 37(2):165–191, 2019. ISSN 0723-0869. doi: https://doi.org/10.1016/j.exmath.2018.01.002. URL https://www.sciencedirect.com/science/article/pii/S0723086918300021.

[6] Camille Bilodeau, Wengong Jin, Tommi Jaakkola, Regina Barzilay, and Klavs F Jensen. Generative models for molecular discovery: Recent advances and challenges. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 12(5):e1608, 2022.

[7] Nathan Brown, Marco Fiscato, Marwin H. S. Segler, and Alain C. Vaucher. Guacamol: Benchmarking models for de novo molecular design. *J. Chem. Inf. Model.*, 59(3):1096–1108, 2019.

[8] Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. In *NeurIPS*, 2022.

[9] Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi S. Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. In *ICML*. OpenReview.net, 2024.

[10] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *CoRR*, abs/1805.11973, 2018.

[11] Ricky T. Q. Chen and Yaron Lipman. Flow matching on general geometries. In *ICLR*. OpenReview.net, 2024.

[12] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *NeurIPS*, pages 6572–6583, 2018.

[13] Xiaohui Chen, Jiaxing He, Xu Han, and Liping Liu. Efficient and degree-guided graph generation via discrete diffusion modeling. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pages 4585–4610. PMLR, 2023.

[14] Hanjun Dai, Azade Nazi, Yujia Li, Bo Dai, and Dale Schuurmans. Scalable deep generative modeling for sparse graphs. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 2302–2312. PMLR, 2020.

[15] Nathaniel Lee Diamant, Alex M. Tseng, Kangway V. Chuang, Tommaso Biancalani, and Gabriele Scalia. Improving graph generation by restricting graph bandwidth. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pages 7939–7959. PMLR, 2023.

[16] D.C Dowson and B.V Landau. The fréchet distance between multivariate normal distributions. *Journal of Multivariate Analysis*, 12(3):450–455, 1982. ISSN 0047-259X. doi: https://doi.org/10.1016/0047-259X(82)90077-X. URL https://www.sciencedirect.com/science/article/pii/0047259X8290077X.

[17] Ian Dunn and David Ryan Koes. Mixed continuous and categorical flow matching for 3d de novo molecule generation. *ArXiv*, pages arXiv–2404, 2024.

[18] Floor Eijkelboom, Grigory Bartosh, Christian Andersson Naesseth, Max Welling, and Jan-Willem van de Meent. Variational flow matching for graph generation. In *NeurIPS*, 2024.

[19] Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. In *NeurIPS*, 2024.

[20] Nikhil Goyal, Harsh Vardhan Jain, and Sayan Ranu. Graphgen: A scalable approach to domain-agnostic labeled graph generation. In *WWW*, pages 1253–1263. ACM / IW3C2, 2020.

[21] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, pages 855–864. ACM, 2016.

[22] Isabel Haasler and Pascal Frossard. Bures-wasserstein means of graphs. In *AISTATS*, volume 238 of *Proceedings of Machine Learning Research*, pages 1873–1881. PMLR, 2024.

[23] Kilian Konstantin Haefeli, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Diffusion models for graphs benefit from discrete state spaces. In *The First Learning on Graphs Conference*, 2022. URL `https://openreview.net/forum?id=CtsKBwhTMKg`.

[24] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

[25] Xiaoyang Hou, Tian Zhu, Milong Ren, Dongbo Bu, Xin Gao, Chunming Zhang, and Shiwei Sun. Improving molecular graph generation with flow matching and optimal transport. *CoRR*, abs/2411.05676, 2024.

[26] John Ingraham, Vikas K. Garg, Regina Barzilay, and Tommi S. Jaakkola. Generative models for graph-based protein design. In *NeurIPS*, pages 15794–15805, 2019.

[27] Keyue Jiang, Bohan Tang, Xiaowen Dong, and Laura Toni. Heterogeneous graph structure learning through the lens of data-generating processes. In *The 28th International Conference on Artificial Intelligence and Statistics*, 2025. URL `https://openreview.net/forum?id=JHK0QBKdYY`.

[28] Jaehyeong Jo, Dongki Kim, and Sung Ju Hwang. Graph generation with diffusion mixture. In *ICML*. OpenReview.net, 2024.

[29] Kacper Kapusniak, Peter Potaptchik, Teodora Reu, Leo Zhang, Alexander Tong, Michael M. Bronstein, Avishek Joey Bose, and Francesco Di Giovanni. Metric flow matching for smooth interpolations on the data manifold. In *NeurIPS*, 2024.

[30] Thomas N. Kipf and Max Welling. Variational graph auto-encoders. *CoRR*, abs/1611.07308, 2016.

[31] Mufei Li, Eleonora Kreacic, Vamsi K. Potluru, and Pan Li. Graphmaker: Can diffusion models generate large attributed graphs? *CoRR*, abs/2310.13833, 2023.

[32] Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, William L. Hamilton, David Duvenaud, Raquel Urtasun, and Richard S. Zemel. Efficient graph generation with graph recurrent attention networks. In *NeurIPS*, pages 4257–4267, 2019.

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

[34] Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky T. Q. Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *CoRR*, abs/2412.06264, 2024.

[35] Chengyi Liu, Wenqi Fan, Yunqing Liu, Jiatong Li, Hang Li, Hui Liu, Jiliang Tang, and Qing Li. Generative diffusion models on graphs: Methods and applications. In *IJCAI*, pages 6702–6711. ijcai.org, 2023.

[36] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *ICLR*. OpenReview.net, 2023.

[37] Karolis Martinkus, Andreas Loukas, Nathanaël Perraudin, and Roger Wattenhofer. SPECTRE: spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 15159–15179. PMLR, 2022.

[38] Robert J. McCann. A convexity principle for interacting gases. *Advances in Mathematics*, 128(1):153–179, 1997. ISSN 0001-8708. doi: https://doi.org/10.1006/aima.1997.1634. URL `https://www.sciencedirect.com/science/article/pii/S0001870897916340`.

[39] Giorgia Minello, Alessandro Bicciato, Luca Rossi, Andrea Torsello, and Luca Cosmo. Generating graphs via spectral diffusion. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=AAXBfJNHDt`.

[40] Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *AISTATS*, volume 108 of *Proceedings of Machine Learning Research*, pages 4474–4484. PMLR, 2020.

[41] I. Olkin and F. Pukelsheim. The distance between two random vectors with given dispersion matrices. *Linear Algebra and its Applications*, 48:257–263, 1982. ISSN 0024-3795. doi: https://doi.org/10.1016/0024-3795(82)90112-4. URL `https://www.sciencedirect.com/science/article/pii/0024379582901124`.

[42] Daniil Polykovskiy, Alexander Zhebrak, Benjamín Sánchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, Artur Kadurin, Sergey I. Nikolenko, Alán Aspuru-Guzik, and Alex Zhavoronkov. Molecular sets (MOSES): A benchmarking platform for molecular generation models. *CoRR*, abs/1811.12823, 2018.

[43] Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky T. Q. Chen. Multisample flow matching: Straightening flows with minibatch couplings. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pages 28100–28127. PMLR, 2023.

[44] Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. Defog: Discrete flow matching for graph generation. *CoRR*, abs/2410.04263, 2024.

[45] Meng Qu, Yoshua Bengio, and Jian Tang. GMNN: graph markov neural networks. In *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 5241–5250. PMLR, 2019.

[46] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

[47] Antoine Siraudin, Fragkiskos D. Malliaros, and Christopher Morris. Cometh: A continuous-time discrete-state graph diffusion model, 2024. URL `https://arxiv.org/abs/2406.06449`.

[48] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*. OpenReview.net, 2021.

[49] Hannes Stärk, Bowen Jing, Chenyu Wang, Gabriele Corso, Bonnie Berger, Regina Barzilay, and Tommi S. Jaakkola. Dirichlet flow matching with applications to DNA sequence design. In *ICML*. OpenReview.net, 2024.

[50] Haoran Sun, Lijun Yu, Bo Dai, Dale Schuurmans, and Hanjun Dai. Score-based continuous-time discrete diffusion models. In *ICLR*. OpenReview.net, 2023.

[51] Asuka Takatsu. On wasserstein geometry of gaussian measures. *Probabilistic approach to geometry*, 57:463–472, 2010.

[52] Ben Taskar, Pieter Abbeel, Ming-Fai Wong, and Daphne Koller. Relational markov networks. *Introduction to statistical relational learning*, 175:200, 2007.

[53] Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Trans. Mach. Learn. Res.*, 2024, 2024.

[54] Clément Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. In *ICLR*. OpenReview.net, 2023.

[55] Clément Vignac, Nagham Osman, Laura Toni, and Pascal Frossard. Midi: Mixed graph and 3d denoising diffusion for molecule generation. In *ECML/PKDD (2)*, volume 14170 of *Lecture Notes in Computer Science*, pages 560–576. Springer, 2023.

[56] C. Villani and American Mathematical Society. *Topics in Optimal Transportation*. Graduate studies in mathematics. American Mathematical Society, 2003. ISBN 9781470418045. URL `https://books.google.co.uk/books?id=MyPjjgEACAAJ`.

[57] Fu-Yun Wang, Ling Yang, Zhaoyang Huang, Mengdi Wang, and Hongsheng Li. Rectified diffusion: Straightness is not your need in rectified flow. *CoRR*, abs/2410.07303, 2024.

[58] Martin Weigt, Robert A. White, Hendrik Szurmant, James A. Hoch, and Terence Hwa. Identification of direct residue contacts in protein–protein interaction by message passing. *Proceedings of the National Academy of Sciences*, 106(1):67–72, 2009. doi: 10.1073/pnas.0805923106. URL `https://www.pnas.org/doi/abs/10.1073/pnas.0805923106`.

[59] Zhe Xu, Ruizhong Qiu, Yuzhong Chen, Huiyuan Chen, Xiran Fan, Menghai Pan, Zhichen Zeng, Mahashweta Das, and Hanghang Tong. Discrete-state continuous-time diffusion for graph generation. *arXiv preprint arXiv:2405.11416*, 2024.

[60] Menglin Yang, Min Zhou, Zhihao Li, Jiahong Liu, Lujia Pan, Hui Xiong, and Irwin King. Hyperbolic graph neural networks: A review of methods and applications. *CoRR*, abs/2202.13852, 2022.

[61] Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 5694–5703. PMLR, 2018.

[62] Meng Yu and Kun Zhan. Bias mitigation in graph diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=CSj72Rr2PB`.

[63] Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. *Semi-supervised learning: From Gaussian fields to Gaussian processes*. School of Computer Science, Carnegie Mellon University, 2003.

[64] Yanqiao Zhu, Yuanqi Du, Yinkai Wang, Yichen Xu, Jieyu Zhang, Qiang Liu, and Shu Wu. A survey on deep graph generation: Methods and applications. In *LoG*, volume 198 of *Proceedings of Machine Learning Research*, page 47. PMLR, 2022.

A Graph Markov random fields: background and theory

A.1 Background of Markov random fields

Markov random fields (MRFs) were originally developed to describe the dynamics of interconnected physical systems such as molecules and proteins [58, 3]. MRFs are energy-based models that have the following probability density:

$$p(\mathcal{X}) = \frac{1}{Z}\prod_c \phi_c\left(x_c\right) = \frac{1}{Z}e^{-E(\mathcal{X})/kT}, \tag{15}$$

where the energy $E(\mathcal{X})$ is used to describe the whole connected system. For instance, in the molecule system that consists of atoms and bonds, the overall energy is decomposed into the atom-wise potential $\varphi_1(v), \forall v \in \mathcal{V}$ and bond-wise potential $\varphi_2(u, v)$, $\forall e_{uv} \in \mathcal{E}$. MRFs serve as a natural and elegant way to describe general graph systems.

The energy-based models have an intrinsic relationship with generative models. As an example, Song et al. [48] derived the relationship between diffusion models and the Langevin dynamics, which is used to describe the evolution of an energy-based model. It is shown that the diffusion models are trying to approximate the score function $\nabla_{\mathcal{X}} \log p(\mathcal{X})$. In the energy-based models, the score function is just the gradient of energy, $\nabla_{\mathcal{X}} \log p(\mathcal{X}) = -\nabla_{\mathcal{X}} E(\mathcal{X})$, and the Langevin dynamics becomes transiting between states with different energies.

The idea of our paper originated from the two facts: MRFs are energy-based model describing connected systems, and the energy-based models have their intrinsic relationship with the diffusion/flow models. Thus, if a model is required to describe the evolution of the whole graph system, we believe it is natural to consider constructing a probability path for two graph distributions with MRFs as the backbone.

A.2 Derivation of graph Markov random fields

We show the derivation of Definition 2, which is restated here:

Definition 3 (Graph Markov Random Fields)**.** GraphMRF statistically describes graphs as,

$$\begin{aligned} &p(\mathcal{G}; \boldsymbol{G}) = p(\mathcal{X}, \mathcal{E}; \boldsymbol{X}, \boldsymbol{W}) = p(\mathcal{X}; \boldsymbol{X}, \boldsymbol{W}) \cdot p(\mathcal{E}; \boldsymbol{W}) \text{ where } \mathcal{E} \sim \delta(\boldsymbol{W}) \text{ and} \\ &\operatorname{vec}(\mathcal{X}) \sim \mathcal{N}\left(\boldsymbol{X}, \boldsymbol{\Lambda}^{\dagger}\right), \text{ with } \boldsymbol{X} = \operatorname{vec}(\boldsymbol{V}^{\dagger}\boldsymbol{\mu}), \boldsymbol{\Lambda} = (\nu\boldsymbol{I} + \boldsymbol{L}) \otimes \boldsymbol{V}^{\top}\boldsymbol{V}. \end{aligned} \tag{16}$$

The $\otimes$ is the Kronecker product, $\operatorname{vec}(\cdot)$ is the vectorization operator and $\boldsymbol{I}$ is the identity matrix.

Derivation:

We start from

$$p(\mathcal{X}; \boldsymbol{X}, \boldsymbol{W}) \propto \prod_v \exp\left\{-(\nu + d_v)\|\boldsymbol{V}x_v - \boldsymbol{\mu}_v\|^2\right\} \prod_{u,v} \exp\left\{w_{uv}\left[(\boldsymbol{V}x_u - \boldsymbol{\mu}_u)^{\top}(\boldsymbol{V}x_v - \boldsymbol{\mu}_v)\right]\right\}. \tag{17}$$

We assume that the linear transformation matrix has dimension $\boldsymbol{V} \in \mathbb{R}^{K' \times K}$ given that $x_v \in \mathbb{R}^K$ and define a transformed variable

$$h_v \equiv \boldsymbol{V}\, x_v - \boldsymbol{\mu}_v \in \mathbb{R}^{K'}, \text{ stacking as } \mathcal{H} \in \mathbb{R}^{|\mathcal{V}| \times K'}. \tag{18}$$

The probability becomes

$$P(\mathcal{H}; \boldsymbol{X}, \boldsymbol{W}) \propto \prod_v \exp\Big\{-(\nu + d_v)\|h_v\|^2\Big\} \prod_{u,v} \exp\Big\{w_{uv}\, h_u^{\top} h_v\Big\}. \tag{19}$$

Then, the terms inside the exponent in Eq. (19) become

$$\begin{aligned} -\sum_v (\nu + d_v)\|h_v\|^2 + \sum_{u,v} w_{uv}\, h_u^{\top} h_v &= -\sum_v (\nu + d_v)\, h_v^{\top} h_v + \sum_{u,v} w_{uv}\, h_u^{\top} h_v \\ &= -\sum_{u,v} h_u^{\top}\Big[(\nu + d_u)\delta_{uv} - w_{uv}\Big] h_v, \end{aligned}$$

where the Kronecker delta $\delta_{uv} = 1$ if $u = v$ and 0 else. We define a squared matrix Λ' to arrange the inner term, which can be written as,

$$\boldsymbol{\Lambda}' = \nu\boldsymbol{I} + \boldsymbol{L} \quad \text{with} \quad \boldsymbol{\Lambda}'_{uv} = (\nu + d_u)\delta_{uv} - w_{uv}. \tag{20}$$

$\boldsymbol{I}$ is the identity matrix. Thus, the exponent in compact matrix form gives

$$-\frac{1}{2}\operatorname{Tr}(\mathcal{H}^\top \boldsymbol{\Lambda}' \mathcal{H}), \text{ where } \mathcal{H} = \begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ h_{|\mathcal{V}|} \end{pmatrix}. \tag{21}$$

It is possible to rearrange the exponent as

$$\operatorname{Tr}(\mathcal{H}^\top \boldsymbol{\Lambda}' \mathcal{H}) = \operatorname{vec}(\mathcal{H})^\top (\boldsymbol{\Lambda} \otimes \boldsymbol{I}) \operatorname{vec}(\mathcal{H}), \tag{22}$$

where $\otimes$ denotes the Kronecker product. This is exactly in the form of a multivariate colored Gaussian. Thus, the joint distribution of $\operatorname{vec}(\mathcal{H})$ (of dimension $|\mathcal{V}|K'$) is given by

$$\operatorname{vec}(\mathcal{H}) \sim \mathcal{N}\Big(0,\ (\nu \boldsymbol{I} + \boldsymbol{L}))^{-1} \otimes \boldsymbol{I}_{K'}\Big), \tag{23}$$

Recall that $h_v = \boldsymbol{V}\, x_v - \boldsymbol{\mu}_v$, we obtain

$$\operatorname{vec}(\mathcal{H}) = (\boldsymbol{I} \otimes \boldsymbol{V})\ \operatorname{vec}(\mathcal{X}) - \operatorname{vec}(\boldsymbol{\mu}). \tag{24}$$

Since the transformation is linear, the distribution over $\mathcal{X}$ remains Gaussian. By the properties of linear transformations of Gaussians, if

$$\operatorname{vec}(\mathcal{H}) \sim \mathcal{N}(\operatorname{vec}(\boldsymbol{\mu}), \boldsymbol{\Sigma}_h), \operatorname{vec}(\mathcal{X}) = (\boldsymbol{I} \otimes \boldsymbol{V}^\dagger)\ \operatorname{vec}(\mathcal{H}), \tag{25}$$

then

$$\operatorname{vec}(\mathcal{X}) \sim \mathcal{N}\Big((\boldsymbol{I} \otimes \boldsymbol{V}^\dagger)\ \operatorname{vec}(\boldsymbol{\mu}),\ (\boldsymbol{I}_n \otimes \boldsymbol{V}^\dagger)\, \boldsymbol{\Sigma}_{\mathcal{H}}\, (\boldsymbol{I}_n \otimes \boldsymbol{V}^\dagger)^\top\Big). \tag{26}$$

Thus, using the mixed-product property of the Kronecker product,

$$(\boldsymbol{I} \otimes \boldsymbol{V}^\dagger)((\nu \boldsymbol{I} + \boldsymbol{L})^{-1} \otimes \boldsymbol{I})(\boldsymbol{I}_n \otimes \boldsymbol{V}^\dagger)^\top = (\boldsymbol{L} + \nu \boldsymbol{I})^{-1} \otimes (\boldsymbol{V}^\dagger \boldsymbol{V}^{\dagger\top}) \tag{27}$$

Finally, the joint distribution over $\mathcal{X}$ is

$$\begin{aligned} &\operatorname{vec}(\mathcal{X}) \sim \mathcal{N}(\boldsymbol{X}, \Sigma), \\ &\text{with } \boldsymbol{X} = (\boldsymbol{I} \otimes \boldsymbol{V}^\dagger)\ \operatorname{vec}(\boldsymbol{\mu}) = \operatorname{vec}(\boldsymbol{V}^\dagger \boldsymbol{\mu}) \\ &\text{and } \Sigma = (\nu \boldsymbol{I} + \boldsymbol{L})^{-1} \otimes \left(\boldsymbol{V}^\dagger \boldsymbol{V}^{\dagger\top}\right), \end{aligned} \tag{28}$$

We use the following lemma:

Lemma 1. *Given two invertible matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, their Kronecker product satisfies $(\boldsymbol{A} \otimes \boldsymbol{B})^{-1} = \boldsymbol{A}^{-1} \otimes \boldsymbol{B}^{-1}$.*

So that we get

$$\operatorname{vec}(\mathcal{X}) \sim \mathcal{N}\left(\boldsymbol{X}, \Lambda^\dagger\right), \text{ with } \boldsymbol{X} = \operatorname{vec}(\boldsymbol{V}^\dagger \boldsymbol{\mu}), \Lambda = (\nu \boldsymbol{I} + \boldsymbol{L}) \otimes \boldsymbol{V}^\top \boldsymbol{V}. \tag{29}$$

which ends the derivation.

A.3 The usage scope of graph Markov random fields

Given that our Graph Markov random fields (GraphMRF) have an explicit form to constrain the graph distribution, it inherits certain inductive biases and we have to properly understand the usage scenarios.

To understand the scenarios which GraphMRF could be used, we start by stating the plain form of MRF by melting the linear transformer matrix and the mean term, i.e., giving $h_v = \boldsymbol{V} x_v - \boldsymbol{\mu}$ which gives the probability density as,

$$P(\mathcal{H} \mid \boldsymbol{L}) \propto \exp(-\operatorname{trace}(\mathcal{H}^\top (\boldsymbol{L} + \nu \boldsymbol{I})\mathcal{H})) = \exp(-\boldsymbol{w}_{uv} \|h_u - h_v\|^2 - \nu \sum_u \|h_u\|_{\mathrm{F}}^2) \tag{30}$$

Plain graph Markov random fields. To understand the scenarios which we can utilize MRF to model graphs, we first consider the simplest case when $\boldsymbol{V}$ is rectangular orthogonal (semi-orthogonal) matrix such that $\boldsymbol{V}^\top \boldsymbol{V} = \boldsymbol{I}$ and the mean $\boldsymbol{\mu} = 0$, the probability density becomes,

$$P(\boldsymbol{X}, \boldsymbol{L}) \propto \exp(-\boldsymbol{X}^\top(\boldsymbol{L} + \nu\boldsymbol{I})\boldsymbol{X}) = \exp(-\sum_{\{u,v\}\in\mathcal{E}} \boldsymbol{W}_{uv}(\boldsymbol{x}_u - \boldsymbol{x}_v)^2 - \nu\sum_u \boldsymbol{x}_u^2) \tag{31}$$

As $\nu \to 0$, the exponent term inside becomes

$$\mathcal{S}(\boldsymbol{X}, \boldsymbol{L}) = -\sum_{\{u,v\}\in\mathcal{E}} \boldsymbol{W}_{uv}(\boldsymbol{x}_u - \boldsymbol{x}_v)^2 = \mathrm{trace}(\boldsymbol{X}^\top \boldsymbol{L}\boldsymbol{X}), \tag{32}$$

where we name $\mathcal{S}(\boldsymbol{X}, \boldsymbol{L})$ as the smoothness of the graph features. The smoothness measures how similar the neighbors connected are. For instance, if there exists an edge between node u and v weighted as $\boldsymbol{W}_{uv}$, the likelihood will be higher if $\boldsymbol{x}_u$ and $\boldsymbol{x}_v$ be similar, so that $\|\boldsymbol{x}_u - \boldsymbol{x}_v\|^2$ are small. This suggests that the probability will be higher if the $\mathcal{S}(\boldsymbol{X}, \boldsymbol{L})$ is small.

This vanilla form captures the first type of graphs the GraphMRF can model - the homophily graphs, i.e., similar nodes (measured by the node attribute) may be more likely to attach to each other than dissimilar ones. This includes the social networks with friendship, traffic networks, etc.

Graph Markov random fields with embeddings. Now we move one step further to consider the graphs with linear transformer matrix $\boldsymbol{V}$. Linear transformation provides a map from the feature space to the latent space, which can be considered as an embedding method to empower the models with better expressiveness. As a simple example, when the $\boldsymbol{V}$ provides a negative projection, the mapping can capture the heterophily relationships, which means the nodes connected are dissimilar.

Coincidently, this aligns well with the famous embedding method Node2Vec as in Grover and Leskovec [21], where the edge weights are proportional to the negative distance, or the inner product of the embeddings. i.e.,

$$\boldsymbol{W}_{uv} \propto \exp(-\|\boldsymbol{V}\boldsymbol{x}_u - \boldsymbol{V}\boldsymbol{x}_v\|_{\mathrm{F}}^2) \tag{33}$$

In [27] it is derived that learning the parameters of MRFs is intrinsically equivalent to learning embeddings similar to Node2Vec. As such, the expressiveness of MRFs are as good as Node2Vec, which grants its usage to molecule graphs, protein interaction networks, social networks, and knowledge graphs. In our paper we make the assumption is that the linear mapping from $\boldsymbol{X}$ the observation is shared. This requirement translates to that the two graphs should have the same embedding space and feature space, which is practical if the reference distribution and data distributions share the same space.

Graphs without features. We wish to emphasize that even though the GraphMRF is constructed under the assumption that graph features exist, it is capable of modeling the non-attributed graphs, such as planar and SBM graphs. To do so, we consider the optimization over the Rayleigh function: It is shown that, if $v_1, \ldots, v_{k-1}$ are orthonormal eigenvectors for $\lambda_1, \ldots, \lambda_{k-1}$, then the eigenvalues satisfy,

$$\lambda_k = \min_{\substack{\boldsymbol{x}\neq 0 \\ \boldsymbol{x}\perp\boldsymbol{v}_1,\ldots,\boldsymbol{v}_{k-1}}} R(\boldsymbol{x}), \text{ with } R(\boldsymbol{x}) = \frac{\boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x}}{\boldsymbol{x}^T\boldsymbol{x}} \tag{34}$$

In such a scenario, the graphs are no longer related to the actual node features, but instead, the eigenvectors $\boldsymbol{v}_k$ serve as an intrinsic graph feature. Interestingly, $R(\boldsymbol{x})$ is the normalized form of the smoothness as in Eq. (31), which is exactly equivalent to the eigenvalues. This means that if the spectrum of the graph Laplacian focuses on the low-frequency components, the corresponding graphs would give a higher probability in terms of MRFs if we view the eigenvectors as the node features. Such a pattern could capture the graph distributions almost all the synthetic datasets, such as Planar, SBM, TLS, COMM20 datasets, satisfy such a property. There is no surprise that our models capture better global patterns in those datasets. But it is also worth pointing out that there exists exceptions, such as the tree graph, which does not have a clear clustering pattern, thus the spectrum does not follow our GraphMRF.

Future work. A limitation of our method is that it cannot easily capture the generation of graphs with multiple relation types, which we name heterogeneous graphs. Even though we utilize an intuitive solution in the experiment to produce Table 2: we first sample the pure graph structure without

edge types to produce the graph backbone, and then sample the edge types via liner interpolated probability on top of the backbone. The solution provides preliminary results for the graph generation in multi-relational graphs, but still requires improvements. Fortunately, there exists a few ways to extend the GraphMRF to heterogeneous graphs [27]. An interesting future work can be generalizing our model to heterogeneous graphs by considering GraphMRF variants, such as the H2MN proposed in Jiang et al. [27].

B Proofs

B.1 Wasserstein Distance between two colored gaussian distributions

We first prove the lemma that captures the Wasserstein distance between two colored Gaussians, which will be used in deriving our Bures-Wasserstein distances in graph generations.

Lemma 2. *Consider two measures $\eta_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0)$ and $\eta_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)$, describing two colored Gaussian distributions with mean $\boldsymbol{\mu}_0, \boldsymbol{\mu}_1$ and covariance matrices Σ_0, Σ_1. Then the Wasserstein distance between these probability distributions is given by*

$$\left(\mathcal{W}_2(\eta_0, \eta_1)\right)^2 = \|\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1\|^2 + \mathrm{Tr}\left(\Sigma_0 + \Sigma_1 - 2\left(\Sigma_0^{1/2}\Sigma_1\Sigma_0^{1/2}\right)^{1/2}\right).$$

Proof. We first state the following proposition.

Proposition 4. *(Translation Invariance of the 2-Wasserstein Distance for Gaussian Measures) Consider two measures $\eta_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \Sigma_0)$ and $\eta_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)$ and their centered measure as $\tilde{\eta}_0 = \mathcal{N}(0, \Sigma_0)$ and $\tilde{\eta}_1 = \mathcal{N}(0, \Sigma_1)$, the squared Wasserstein distance decomposes as*

$$\mathcal{W}_2^2(\eta_0, \eta_1) = \|\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1\|_2^2 + \mathcal{W}_2^2(\tilde{\eta}_0, \tilde{\eta}_1)$$

Proof:

Consider two random vectors $\mathcal{X}, \mathcal{Y}$ distributed as η_0, η_1,

$$\mathcal{X} = \boldsymbol{\mu}_0 + \tilde{\mathcal{X}}, \mathcal{Y} = \mu_1 + \tilde{\mathcal{Y}}, \text{ with } \tilde{\mathcal{X}} \sim \tilde{\eta}_0, \tilde{\mathcal{Y}} \sim \tilde{\eta}_1.$$

For any coupling $(\mathcal{X}, \mathcal{Y})$, we consider the expected squared Euclidean distance,

$$\begin{aligned} \mathbb{E}_{\mathcal{X},\mathcal{Y}}\|\mathcal{X} - \mathcal{Y}\|^2 &= \mathbb{E}_{\mathcal{X},\mathcal{Y}}\left\|\boldsymbol{mu}_0 - \boldsymbol{\mu}_1 + (\tilde{\mathcal{X}} - \tilde{\mathcal{Y}})\right\|^2. \\ &= \|\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1\|^2 + 2\left\langle \boldsymbol{\mu}_0 - \mu_1, \tilde{\mathcal{X}} - \tilde{\mathcal{Y}}\right\rangle + \mathbb{E}_{\tilde{\mathcal{X}},\tilde{\mathcal{Y}}}\|\tilde{\mathcal{X}} - \tilde{\mathcal{Y}}\|^2 \end{aligned} \tag{35}$$

Since $\tilde{\mathcal{X}}$ and $\tilde{\mathcal{Y}}$ both have zero mean, we have $\mathbb{E}[\tilde{\mathcal{X}} - \tilde{\mathcal{Y}}] = 0$ so the cross-term vanishes. Thus,

$$\mathbb{E}\|\mathcal{X} - \mathcal{Y}\|^2 = \|\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1\|^2 + \mathbb{E}\|\tilde{\mathcal{X}} - \tilde{\mathcal{Y}}\|^2 \tag{36}$$

Take the definition of 2-Wasserstein distance, the infimum over all couplings directly yields

$$\begin{aligned} \left(\mathcal{W}_2(\eta_0, \eta_1)\right)^2 &= \inf_{\pi \in \Pi(\eta_0,\eta_1)} \int \|\mathcal{X} - \mathcal{Y}\|^2 \, d\pi(\mathcal{X}, \mathcal{Y}). \\ &= \|\boldsymbol{\mu}_0 - \boldsymbol{\mu}_1\|^2 + \mathcal{W}_2^2(\tilde{\eta}_0, \tilde{\eta}_1) \end{aligned} \tag{37}$$

This completes the proof of Proposition 4.

Now we prove the flowing proposition, which will give us our lemma.

Proposition 5. *Given two centered measures as $\tilde{\eta}_0 = \mathcal{N}(0, \Sigma_0)$ and $\tilde{\eta}_1 = \mathcal{N}(0, \Sigma_1)$*

$$\mathcal{W}_2^2(\tilde{\eta}_0, \tilde{\eta}_1) = \mathrm{Tr}\left(\Sigma_0 + \Sigma_1 - 2\left(\Sigma_1^{1/2}\Sigma_0\Sigma_1^{1/2}\right)^{1/2}\right). \tag{38}$$

proof. The coupling π of $\tilde{\eta}_0$ and $\tilde{\eta}_1$ is a joint Gaussian measure with zero mean and covariance matrix

$$\Sigma_c = \begin{pmatrix} \Sigma_0 & \boldsymbol{C} \\ \boldsymbol{C}^T & \Sigma_1 \end{pmatrix} \succeq 0, \tag{39}$$

where $\boldsymbol{C}$ is the cross-covariance and $\succeq$ means the matrix is positive semi-definitive (PSD). The expected squared distance between the two random vectors $(\mathcal{X}, \mathcal{Y})$ drawn from π is then described as,

$$\begin{aligned}\mathbb{E}\|\mathcal{X}-\mathcal{Y}\|^2 &= \operatorname{Tr}(\mathbb{E}[(\mathcal{X}-\mathcal{Y})(\mathcal{X}-\mathcal{Y})^\top]) \\ &= \operatorname{Tr}(\Sigma_0) + \operatorname{Tr}(\Sigma_1) - 2\operatorname{Tr}(\boldsymbol{C}).\end{aligned} \tag{40}$$

The definition of Wasserstein distance gives,

$$\mathcal{W}_c(\eta_0, \eta_1) = \inf_{\pi \in \Pi(\eta_0, \eta_1)} \mathbb{E}\|\mathcal{X}-\mathcal{Y}\|^2 \tag{41}$$

Thus, minimizing the Wasserstein distance is equivalent to maximizing $\operatorname{Tr}(\boldsymbol{C})$ over all $\boldsymbol{C}$ subject to the joint covariance is positive semi-definite (PSD). It turns out (see Dowson and Landau [16], Olkin and Pukelsheim [41], Takatsu [51]) that the condition in Eq. (39) is equivalent to,

$$\Sigma_1 - \boldsymbol{C}^\top \Sigma_0^{-1} \boldsymbol{C} \succeq 0 \leftrightarrow \Sigma_0^{-1/2} \boldsymbol{C} \Sigma_1^{-1/2} \text{ has operator norm } \leq 1 \tag{42}$$

So we denote $\boldsymbol{K} \coloneqq \Sigma_0^{-1/2} \boldsymbol{C} \Sigma_1^{-1/2}$ with $\|\boldsymbol{K}\|_{\text{op}} \leq 1$. Then

$$\operatorname{Tr}(\boldsymbol{C}) = \operatorname{Tr}\left(\Sigma_0^{1/2} \boldsymbol{K} \Sigma_1^{1/2}\right) = \operatorname{Tr}\left(\boldsymbol{K} \Sigma_1^{1/2} \Sigma_0^{1/2}\right).$$

Using von Neumann trace inequality, its trace inner-product with $\boldsymbol{K}$ is maximized by choosing $\boldsymbol{K} = \boldsymbol{I}$ on the support.

$$\max_{\|\boldsymbol{K}\|_{op} \leq 1} \operatorname{Tr}(\boldsymbol{K}\boldsymbol{A}) = \operatorname{Tr}\left(\boldsymbol{M}^{1/2}\right), \quad \boldsymbol{M} = \sqrt{\boldsymbol{A}\boldsymbol{A}^\top} = \Sigma_1^{1/2} \Sigma_0 \Sigma_1^{1/2}$$

Hence the optimal value of $\operatorname{Tr}(\boldsymbol{C})$ is

$$\operatorname{Tr}(\boldsymbol{C}^*) = \operatorname{Tr}\left[\left(\Sigma_1^{1/2} \Sigma_0 \, \Sigma_1^{1/2}\right)^{1/2}\right]$$

Substituting this optimal value into the expression of Wasserstein distance, we obtain

$$\mathcal{W}_2^2\left(\tilde{\eta}_0, \tilde{\eta}_1\right) = \operatorname{Tr}(\Sigma_0) + \operatorname{Tr}(\Sigma_1) - 2\operatorname{Tr}\left[\left(\Sigma_1^{1/2} \Sigma_0 \, \Sigma_1^{1/2}\right)^{1/2}\right]. \tag{43}$$

This completes the proof of proposition 5. Taking Proposition 4 and Proposition 5 together, we proved Lemma 2.

B.2 Derivation of the graph Wasserstein distance under MRF

We then prove the Bures-Wasserstein distance for two graph distributions. We restate Proposition 1,

Proposition 6 (Bures-Wasserstein Distance)**.** *Consider two same-sized graphs $\mathcal{G}_0 \sim p\left(\mathcal{X}_0, \mathcal{E}_0\right)$ and $\mathcal{G}_1 \sim p\left(\mathcal{X}_1, \mathcal{E}_1\right)$ with $\boldsymbol{V}$ shared for two graphs, described by the distribution in Definition 2. When the graphs are equipped with graph Laplacian matrices $\boldsymbol{L}_0$ and $\boldsymbol{L}_1$ satisfying 1) is Positive Semi-Definite (PSD) and 2) has only one zero eigenvalue. The Bures-Wasserstein distance between these two random graph distributions is given by*

$$d_{BW}(\mathcal{G}_0, \mathcal{G}_1) = \|\boldsymbol{X}_0 - \boldsymbol{X}_1\|_F^2 + \beta \operatorname{Tr}\left(\boldsymbol{L}_0^\dagger + \boldsymbol{L}_1^\dagger - 2\left(\boldsymbol{L}_0^{\dagger/2} \boldsymbol{L}_1^\dagger \boldsymbol{L}_0^{\dagger/2}\right)^{1/2}\right), \tag{44}$$

as $\nu \to 0$ and β is a constant related to the norm of $\boldsymbol{V}$.

Specifically, Definition 2 uses graph Markov random fields to describe a graph as

$$\begin{aligned}&p(\mathcal{G}; \boldsymbol{G}) = p(\mathcal{X}, \mathcal{E}; \boldsymbol{X}, \boldsymbol{W}) = p(\mathcal{X}; \boldsymbol{X}, \boldsymbol{W}) \cdot p(\mathcal{E}; \boldsymbol{W}) \text{ where } \mathcal{E} \sim \delta(\boldsymbol{W}) \text{ and} \\ &\operatorname{vec}(\mathcal{X}) \sim \mathcal{N}\left(\boldsymbol{X}, \boldsymbol{\Lambda}^\dagger\right), \text{ with } \boldsymbol{X} = \operatorname{vec}(\boldsymbol{V}^\dagger \boldsymbol{\mu}), \boldsymbol{\Lambda} = (\nu \boldsymbol{I} + \boldsymbol{L}) \otimes \boldsymbol{V}^\top \boldsymbol{V}.\end{aligned} \tag{45}$$

With the graph Wasserstein distance defined as,

$$\text{(Graph Wasserstein Distance)} \quad d_{\text{BW}}(\mathcal{G}_0, \mathcal{G}_1) \coloneqq \mathcal{W}_c\left(\eta_{\mathcal{G}_0}, \eta_{\mathcal{G}_1}\right) = \mathcal{W}_c(\eta_{\mathcal{X}_0}, \eta_{\mathcal{X}_1}) + \mathcal{W}_c(\eta_{\mathcal{E}_0}, \eta_{\mathcal{E}_1}).$$

We first consider calculating $\mathcal{W}_c(\eta_{\mathcal{X}_0}, \eta_{\mathcal{X}_1})$. Specifically, this is the distance between two colored Gaussian measures where

$$\begin{aligned}&\eta_i \sim \mathcal{N}\left(\boldsymbol{\mu}_i', \Sigma_i\right), \quad i = 0, 1, \\ &\text{where } \boldsymbol{\mu}_i' = \boldsymbol{V}_i \otimes \boldsymbol{\mu}_i \text{ and } \Sigma_i^{-1} = \Lambda_i = (\nu \boldsymbol{I} + \boldsymbol{L}_i) \otimes (\boldsymbol{V}_i^\top \boldsymbol{V}_i).\end{aligned} \tag{46}$$

where we first assume that these two Gaussians are emitted from different linear transformation matrices $\boldsymbol{V}_0$ and $\boldsymbol{V}_1$. This will bring us the most general and flexible form that could be universally applicable, and potentially can bring more insights to future work. Next, we will inject a few assumptions to arrive at a more practical form for building the flow matching models.

An important property of Kronecker product: Given two invertible matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, their Kronecker product satisfies $(\boldsymbol{A} \otimes \boldsymbol{B})^{-1} = \boldsymbol{A}^{-1} \otimes \boldsymbol{B}^{-1}$. Using such a property, in the limit as $\nu \to 0$, we have

$$\Lambda_i \to \boldsymbol{L}_i \otimes (\boldsymbol{V}_i^\top \boldsymbol{V}_i) \quad \Longrightarrow \quad \Sigma_i = \boldsymbol{L}_i^{-1} \otimes (\boldsymbol{V}_i^\top \boldsymbol{V}_i)^{-1}. \tag{47}$$

According to Propensity 2, the squared 2-Wasserstein distance between two Gaussian measures is given by

$$\mathcal{W}_2^2(\eta_0, \eta_1) = \underbrace{\|\boldsymbol{\mu}_0' - \boldsymbol{\mu}_1'\|^2}_{\text{Mean term}} + \underbrace{\mathrm{Tr}\Big(\Sigma_0 + \Sigma_1 - 2\Big(\Sigma_0^{1/2}\Sigma_1\,\Sigma_0^{1/2}\Big)^{1/2}\Big)}_{\text{Covariance Term}}. \tag{48}$$

Mean Term. Since $\boldsymbol{\mu}_i' = \boldsymbol{V} \otimes \boldsymbol{\mu}_i$, the mean difference becomes

$$\|\boldsymbol{\mu}_0' - \boldsymbol{\mu}_1'\|^2 = \|\boldsymbol{V}_0\boldsymbol{\mu}_0 - \boldsymbol{V}_1\boldsymbol{\mu}_1\|_F^2 = \|\boldsymbol{X}_0 - \boldsymbol{X}_1\|_{\mathrm{F}}^2 \tag{49}$$

Covariance term. Using the property of the Kronecker product, the square root of Eq. (47) factors in as

$$\Sigma_i^{1/2} = \boldsymbol{L}_i^{-1/2} \otimes (\boldsymbol{V}_i^\top \boldsymbol{V}_i)^{-1/2}. \tag{50}$$

and

$$\Sigma_0^{1/2}\Sigma_1\,\Sigma_0^{1/2} = \Big(\boldsymbol{L}_0^{-1/2}\boldsymbol{L}_1^{-1}\boldsymbol{L}_0^{-1/2}\Big) \otimes \Big((\boldsymbol{V}_0^\top \boldsymbol{V}_0)^{-1/2}(\boldsymbol{V}_1^\top \boldsymbol{V}_1)^{-1}(\boldsymbol{V}_0^\top \boldsymbol{V}_0)^{-1/2}\Big) \tag{51}$$

We first look into the term related to $\boldsymbol{V}_0$ and $\boldsymbol{V}_1$, which is,

$$\begin{aligned}\mathrm{Tr}\Big((\boldsymbol{V}_0^\top \boldsymbol{V}_0)^{-1/2}(\boldsymbol{V}_1^\top \boldsymbol{V}_1)^{-1}(\boldsymbol{V}_0^\top \boldsymbol{V}_0)^{-1/2}\Big) &= \mathrm{Tr}\Big((\boldsymbol{V}_1^\top \boldsymbol{V}_1)^{-1}(\boldsymbol{V}_0^\top \boldsymbol{V}_0)^{-1/2}(\boldsymbol{V}_0^\top \boldsymbol{V}_0)^{-1/2}\Big) \\ &= \mathrm{Tr}\Big((\boldsymbol{V}_1^\top \boldsymbol{V}_1)^{-1}(\boldsymbol{V}_0^\top \boldsymbol{V}_0)^{-1}\Big)\end{aligned} \tag{52}$$

As $\mathrm{Tr}(\boldsymbol{A} + \boldsymbol{B}) = \mathrm{Tr}(\boldsymbol{A}) + \mathrm{Tr}(\boldsymbol{B})$ the covariance term becomes

Covariance Term

$$\begin{aligned}&= \mathrm{Tr}\Big(\Sigma_0 + \Sigma_1 - 2\Big(\Sigma_0^{1/2}\Sigma_1\,\Sigma_0^{1/2}\Big)^{1/2}\Big) \\ &= \mathrm{Tr}(\Sigma_0) + \mathrm{Tr}(\Sigma_1) - 2\,\mathrm{Tr}\Big((\Sigma_0^{1/2}\Sigma_1\,\Sigma_0^{1/2})^{1/2}\Big) \\ &= \mathrm{Tr}\Big(\boldsymbol{L}_0^{-1} \otimes (\boldsymbol{V}_0^\top \boldsymbol{V}_0)^{-1} + \boldsymbol{L}_1^{-1} \otimes (\boldsymbol{V}_1^\top \boldsymbol{V}_1)^{-1} - 2\Big(\boldsymbol{L}_0^{-1/2}\boldsymbol{L}_1^{-1}\boldsymbol{L}_0^{-1/2}\Big)^{1/2} \otimes (\boldsymbol{V}_1^\top \boldsymbol{V}_1)^{-1/2}(\boldsymbol{V}_0^\top \boldsymbol{V}_0)^{-1/2}\Big)\end{aligned} \tag{53}$$

Given that $\mathrm{Tr}(\boldsymbol{A} \otimes \boldsymbol{B}) = \mathrm{Tr}(\boldsymbol{A})\,\mathrm{Tr}(\boldsymbol{B})$ and $\mathrm{Tr}(\boldsymbol{V}^\top \boldsymbol{V}) = \|\boldsymbol{V}\|_{\mathrm{F}}^2$ for any real-valued matrix $\boldsymbol{V}$, we can further derive,

$$\begin{aligned}\text{Covariance Term} = {} & \mathrm{Tr}[(\boldsymbol{V}_0^\top \boldsymbol{V}_0)^{-1}]\,\mathrm{Tr}(\boldsymbol{L}_0^\dagger) + \mathrm{Tr}[(\boldsymbol{V}_1^\top \boldsymbol{V}_1)^{-1}]\,\mathrm{Tr}(\boldsymbol{L}_1^\dagger) \\ & - 2\,\mathrm{Tr}\Big(\boldsymbol{L}_0^{\dagger/2}\boldsymbol{L}_1^\dagger\boldsymbol{L}_0^{\dagger/2}\Big)^{1/2} \cdot \mathrm{Tr}[(\boldsymbol{V}_1^\top \boldsymbol{V}_1)^{-1/2}(\boldsymbol{V}_0^\top \boldsymbol{V}_0)^{-1/2}].\end{aligned} \tag{54}$$

Unfortunately, to simplify this equation, we have to make the two gram matrix, $(\boldsymbol{V}_0^\top \boldsymbol{V}_0)^{-1}$ and $(\boldsymbol{V}_1^\top \boldsymbol{V}_1)^{-1}$ agree, i.e., $(\boldsymbol{V}_1^\top \boldsymbol{V}_1)^{-1} = (\boldsymbol{V}_0^\top \boldsymbol{V}_0)^{-1}$. This will be satisfied if and only if there exists an orthogonal matrix $\boldsymbol{Q}$ such that

$$\boldsymbol{V}_1^\dagger = \boldsymbol{V}_0^\dagger \boldsymbol{Q}.$$

Thus, to further process, we simply consider the case when $\boldsymbol{V}_1$ and $\boldsymbol{V}_0$ are exactly the same, i.e., $\boldsymbol{V}_1 = \boldsymbol{V}_0 = \boldsymbol{V}$ (we have already discussed how realistic this assumption is in Appendix A.3). So that we work under the assumptions that $\|\boldsymbol{V}_0^\dagger\|_F^2 = \|\boldsymbol{V}_1^\dagger\|_F^2 = \beta$, which simplify the trace as

$$\text{Covariance Term} = \beta \cdot \mathrm{Tr}\Big(\boldsymbol{L}_0^\dagger + \boldsymbol{L}_1^\dagger - 2\Big(\boldsymbol{L}_0^{\dagger/2}\boldsymbol{L}_1^\dagger\boldsymbol{L}_0^{\dagger/2}\Big)^{1/2}\Big). \tag{55}$$

Combining the mean term and the covariance term, we obtain the Wasserstein distance of $\mathcal{W}_c(\eta_{\mathcal{X}_0}, \eta_{\mathcal{X}_1})$

For calculating $\mathcal{W}_c(\eta_{\mathcal{E}_0}, \eta_{\mathcal{E}_1})$, we have the freedom to choose the cost function when obtaining the Wasserstein distance. Note that $\boldsymbol{W}$ serves as the prior for the Gaussian covariance matrix Σ, where the covariance has to be positive-semi definite. Thus, according to [5], a proper distance between two positive semi-definite matrices is measured by

$$\mathcal{W}(\eta_{\mathcal{E}_0}, \eta_{\mathcal{E}_1}) = \left\| \Sigma_0^{1/2} - \Sigma_1^{1/2} \right\|_F^2. \tag{56}$$

Coincidently, this is another usage case when the Bures-Wasserstein metric is utilized. Putting everything together, the Wasserstein distance in the limit $\nu \to 0$ is

$$\begin{aligned} d_{\mathrm{BW}}(\mathcal{G}_0, \mathcal{G}_1) &= \|\boldsymbol{V}_0\boldsymbol{\mu}_0 - \boldsymbol{V}_1\boldsymbol{\mu}_1\|_F^2 + (\beta+1) \cdot \mathrm{Tr}\Big(\boldsymbol{L}_0^\dagger + \boldsymbol{L}_1^\dagger - 2\Big(\boldsymbol{L}_0^{\dagger/2}\boldsymbol{L}_1^\dagger\boldsymbol{L}_0^{\dagger/2}\Big)^{1/2}\Big). \\ &= \underbrace{\|\boldsymbol{X}_0 - \boldsymbol{X}_1\|_F^2}_{d_{\boldsymbol{X}}(\boldsymbol{X}_0, \boldsymbol{X}_1)} + (\beta+1) \underbrace{\cdot \mathrm{Tr}\Big(\boldsymbol{L}_0^\dagger + \boldsymbol{L}_1^\dagger - 2\Big(\boldsymbol{L}_0^{\dagger/2}\boldsymbol{L}_1^\dagger\boldsymbol{L}_0^{\dagger/2}\Big)^{1/2}\Big)}_{d_{\boldsymbol{L}}(\boldsymbol{L}_0, \boldsymbol{L}_1)}. \end{aligned} \tag{57}$$

This expression separates the contribution of the mean difference (transformed by $\boldsymbol{V}$) and the discrepancy between the covariance structures (encoded in $\boldsymbol{L}_0$ and $\boldsymbol{L}_1$). This could be further used to derive BW interpolation, which we will show in Appendix C.1. In the main body, constant β actually corresponds to $\beta + 1$ here. This complete our derivation in Proposition 1.

C Derivation of Bures-Wasserstein flow matching

In order to build the flow matching framework, we need to derive the optimal interpolation and the corresponding velocities for the probability path $p(G_t \mid G_0, G_1)$. This is achieved via the OT displacement between two graph distributions.

C.1 The Bures-Wasserstein graph interpolation

We aim to recover the proposition stated as follows.

Proposition 7 (Bures-Wasserstein interpolation)**.** *The graph minimizer of Eq.* (10)*, $\mathcal{G}_t = \{\mathcal{V}, \mathcal{E}_t, \mathcal{X}_t\}$, have its node features following a colored Gaussian distribution, $\mathcal{X}_t \sim \mathcal{N}(\boldsymbol{X}_t, \boldsymbol{\Lambda}_t^\dagger)$ with $\boldsymbol{\Lambda}_t = (\nu\boldsymbol{I} + \boldsymbol{L}_t) \otimes \boldsymbol{V}^\top\boldsymbol{V}$ and edges following $\mathcal{E}_t \sim \delta(\boldsymbol{W}_t)$, specifically,*

$$\boldsymbol{L}_t^\dagger = \boldsymbol{L}_0^{1/2}\left((1-t)\boldsymbol{L}_0^\dagger + t\left(\boldsymbol{L}_0^{\dagger/2}\boldsymbol{L}_1^\dagger\boldsymbol{L}_0^{\dagger/2}\right)^{1/2}\right)^2\boldsymbol{L}_0^{1/2}, \quad \boldsymbol{X}_t = (1-t)\boldsymbol{X}_0 + t\boldsymbol{X}_1 \tag{58}$$

The interpolation is an extension of the concept of mean, where in the optimal transport world, the Wasserstein barycenter (mean) of measures $\eta_0, \ldots \eta_{m-1}$ under weights $\lambda_0, \ldots \lambda_{m-1}$ can be derived over the following optimization problem:

$$\bar{\eta} = \arg\min_{\eta} \sum_{j=0}^{m-1} \lambda_j \left(\mathcal{W}_2\left(\eta, \eta_j\right)\right)^2 \tag{59}$$

When $m = 2$, based on the Bures-Wasserstein (BW) distance, we can define the OT displacement minimization problem on graphs described as,

$$\mathcal{G}_t = \arg\min_{\tilde{\mathcal{G}}} (1-t) d_{\mathrm{BW}}(\mathcal{G}_0, \tilde{\mathcal{G}}) + t d_{\mathrm{BW}}(\tilde{\mathcal{G}}, \mathcal{G}_1). \tag{60}$$

where $d_{\mathrm{BW}}(\mathcal{G}_0, \mathcal{G}_1)$ is described in Proposition 1. The optimal graph interpolation is the solution to the problem.

In the setting of graph, this becomes a two-variable optimization problem, where

$$\mathcal{X}_t, \mathcal{E}_t = \arg\min_{\tilde{\mathcal{X}}, \tilde{\mathcal{E}}} (1-t) d_{\mathrm{BW}}(\mathcal{G}_0, \tilde{\mathcal{G}}) + t d_{\mathrm{BW}}(\tilde{\mathcal{G}}, \mathcal{G}_1). \tag{61}$$

Fortunately, recall in Eq. (57) that our distance measurement $d_{\mathrm{BW}}(\mathcal{G}_0, \mathcal{G}_1)$ is decomposed into $d_{\boldsymbol{X}}(\boldsymbol{X}_0, \boldsymbol{X}_1)$ and $d_{\boldsymbol{L}}(\boldsymbol{L}_0, \boldsymbol{L}_1)$, then the optimization over node and edges are disentangleable into solving the two sub optimization problem,

$$\begin{aligned} &\text{Sub-question 1:} \quad \bar{\boldsymbol{X}}_t = \arg\min_{\tilde{\boldsymbol{X}}} \quad (1-t)\|\boldsymbol{X}_0 - \tilde{\boldsymbol{X}}\|_F^2 + t\|\tilde{\boldsymbol{X}} - \boldsymbol{X}_1\|_F^2 \\ &\text{Sub-question 2:} \quad \bar{\boldsymbol{L}}_t = \arg\min_{\tilde{\boldsymbol{L}}} \quad (1-t)d_{\boldsymbol{L}}(\boldsymbol{L}_0, \tilde{\boldsymbol{L}}) + td_{\boldsymbol{L}}(\boldsymbol{L}_1, \tilde{\boldsymbol{L}}) \end{aligned} \tag{62}$$

This two problems are completely disentangled thus we can solve them separately.

Sub-question 1 For the first problem, we simply set the derivate to 0 and get,

$$(1-t)(\tilde{\boldsymbol{X}} - \boldsymbol{X}_0) + t(\tilde{\boldsymbol{X}} - \boldsymbol{X}_1) = 0 \to \boldsymbol{X}_t = (1-t)\boldsymbol{X}_0 + t(\boldsymbol{X}_1) \tag{63}$$

Subquestion 2 The second subproblem is equivalent in deriving the covariance of Bures-Wasserstein interpolation between two Gaussian measures, $\eta_0 \sim \mathcal{N}\left(0, \boldsymbol{L}_0^\dagger\right)$ and $\eta_1 \sim \mathcal{N}\left(0, \boldsymbol{L}_1^\dagger\right)$. This problem has been properly addressed in Haasler and Frossard [22] and here we just verbose their results. For more details we refer the reader to Haasler and Frossard [22] for a further discussion.

The optimal transport geodesic between $\eta_0 \sim \mathcal{N}\left(0, \boldsymbol{L}_0^\dagger\right)$ and $\eta_1 \sim \mathcal{N}\left(0, \boldsymbol{L}_1^\dagger\right)$ is defined by $\eta_t = ((1-t)I + t\boldsymbol{T})_{\#}\eta_0$, where the symbol "#" denotes the push-forward of a measure by a mapping, $\boldsymbol{T}$ is a linear map that satisfies $\boldsymbol{T}\boldsymbol{L}_0^\dagger\boldsymbol{T} = \boldsymbol{L}_1^\dagger$.

We define a new matrix $\boldsymbol{M}$ and do normalization, which leads to,

$$\boldsymbol{T} = \boldsymbol{L}_0^{1/2}\boldsymbol{M}\boldsymbol{L}_0^{1/2} \tag{64}$$

Plug in gives,

$$\begin{aligned} \boldsymbol{T}\boldsymbol{L}_0^\dagger\boldsymbol{T}^\top &= \boldsymbol{L}_0^{1/2}\boldsymbol{M}\boldsymbol{L}_0^{1/2}\boldsymbol{L}_0^\dagger\left(\boldsymbol{L}_0^{1/2}\boldsymbol{M}\boldsymbol{L}_0^{1/2}\right)^\top \\ &= \boldsymbol{L}_0^{1/2}\boldsymbol{M}\boldsymbol{M}^\top\boldsymbol{L}_0^{1/2}. \end{aligned} \tag{65}$$

So that we obtain

$$\boldsymbol{L}_1^\dagger = \boldsymbol{L}_0^{1/2}\boldsymbol{M}\boldsymbol{M}^\top\boldsymbol{L}_0^{1/2} \to \boldsymbol{M} = (\boldsymbol{L}_0^{\dagger/2}\boldsymbol{L}_1^\dagger\boldsymbol{L}_0^{\dagger/2})^{1/2} \tag{66}$$

Replace $\boldsymbol{T}$ and we get,

$$\boldsymbol{T} = \boldsymbol{L}_0^{1/2}\left(\boldsymbol{L}_0^{\dagger/2}\boldsymbol{L}_1^\dagger\boldsymbol{L}_0^{\dagger/2}\right)^{1/2}\boldsymbol{L}_0^{1/2} \tag{67}$$

Given that the geodesic $\eta_t = ((1-t)\boldsymbol{I} + t\boldsymbol{T})_{\#}\eta_0$ which also has a Gaussian form $\eta_t \sim \mathcal{N}(0, \Sigma_t)$, We can then write the covariance matrix and obtain

$$\begin{aligned} \boldsymbol{L}_t^\dagger = \Sigma_t &= ((1-t)I + t\boldsymbol{T})\boldsymbol{L}_0^\dagger((1-t)I + t\boldsymbol{T}) \\ &= \boldsymbol{L}_0^{1/2}\left((1-t)\boldsymbol{L}_0^\dagger + t\left(\boldsymbol{L}_0^{\dagger/2}\boldsymbol{L}_1^\dagger\boldsymbol{L}_0^{\dagger/2}\right)^{1/2}\right)\boldsymbol{L}_0^{1/2}\boldsymbol{L}_0^\dagger\boldsymbol{L}_0^{1/2}\left((1-t)\boldsymbol{L}_0^\dagger + t\left(\boldsymbol{L}_0^{\dagger/2}\boldsymbol{L}_1^\dagger\boldsymbol{L}_0^{\dagger/2}\right)^{1/2}\right)\boldsymbol{L}_0^{1/2} \\ &= \boldsymbol{L}_0^{1/2}\left((1-t)\boldsymbol{L}_0^\dagger + t\left(\boldsymbol{L}_0^{\dagger/2}\boldsymbol{L}_1^\dagger\boldsymbol{L}_0^{\dagger/2}\right)^{1/2}\right)^2\boldsymbol{L}_0^{1/2} \end{aligned} \tag{68}$$

Which ends the derivation.

Remark 1: Even though the GraphMRF in Definition 2 does rely on an implicit linear emission matrices $\boldsymbol{V}$, the BW interpolation in Proposition 2 can be obtained without explicitly accessing to the $\boldsymbol{V}$ matrices. The property was attractive as in practice we can construct the probability path without explicitly fitting a $\boldsymbol{V}$ beforehand.

C.2 Deriving the velocity of BW interpolation

We first show the general form of the velocity term for the Gaussian and Dirac measures.

Gaussian measure. For a time-parametrized Gaussian density $p_t(x) = \mathcal{N}(x; \boldsymbol{\mu}_t, \Sigma_t)$, the velocity field $v_t(x)$ satisfies the continuity equation

$$\partial_t p_t + \nabla \cdot (p_t v_t) = 0,$$

is an affine function of x. And the instantaneous velocity field follows,

$$v_t(\mathcal{X}) = \dot{\boldsymbol{\mu}}_t + \frac{1}{2}\dot{\Sigma}_t\Sigma_t^{-1}\left(\mathcal{X} - \boldsymbol{\mu}_t\right).$$

Dirac measure. When the measure is a Dirac function,

$$p_t(x) = \delta(\cdot, \boldsymbol{\mu}_t).$$

We can just consider it as the limited case of the Gaussian measure, when $\Sigma_t \to 0$. So that the velocity at simply takes

$$v_t(x) = \dot{\boldsymbol{\mu}}_t.$$

We then move to prove the following propensity for Bures-wasserstein velocity.

Proposition 8 (Bures-Wasserstein velocity)**.** *For the graph $\mathcal{G}_t$ following BW interpolation in Proposition 2, the conditional velocity at time t with observation G_t is given as,*

$$\begin{aligned} &v_t(E_t \mid G_0, G_1) = \dot{\boldsymbol{W}}_t = diag(\dot{\boldsymbol{L}}_t) - \dot{\boldsymbol{L}}_t, \quad v_t(X_t \mid G_0, G_1) = \frac{1}{1-t}(\boldsymbol{X}_1 - \boldsymbol{X}_t) \\ &\text{with } \dot{\boldsymbol{L}}_t = 2\boldsymbol{L}_t - \boldsymbol{T}\boldsymbol{L}_t - \boldsymbol{L}_t\boldsymbol{T} \text{ and } \boldsymbol{T} = \boldsymbol{L}_0^{1/2}(\boldsymbol{L}_0^{\dagger/2}\boldsymbol{L}_1^{\dagger}\boldsymbol{L}_0^{\dagger/2})^{1/2}\boldsymbol{L}_0^{1/2} \end{aligned} \tag{69}$$

where $\boldsymbol{W}_t = \boldsymbol{D}_t - \boldsymbol{L}_t$ and $\boldsymbol{L}_t$ defined in Eq. (11)*.*

Proof:

The graph structure velocity. As we assume the edges, $E_t \sim \delta(\cdot, \boldsymbol{W}_t)$, following a dirac distribution, the velocity is defined as

$$v_t(E_t) = \dot{\boldsymbol{W}}_t.$$

Given that, $\dot{\boldsymbol{W}}_t = \text{diag}(\dot{\boldsymbol{L}}_t) - \dot{\boldsymbol{L}}_t$, we transit fo deriving the derivative of the Laplacian matrix, $\dot{\boldsymbol{L}}_t$. Using the fact that,

$$\frac{d}{dt}\left(\boldsymbol{A}^{-1}\right) = -\boldsymbol{A}^{-1}\frac{d\boldsymbol{A}}{dt}\boldsymbol{A}^{-1}$$

we obtain the derivate of Laplacian matrix,

$$\dot{\boldsymbol{L}}_t = \frac{d(\Sigma_t^{\dagger})}{dt} = \Sigma_t^{\dagger}\frac{d\Sigma_t}{dt}\Sigma_t^{\dagger} = \boldsymbol{L}_t\frac{d\Sigma_t}{dt}\boldsymbol{L}_t \tag{70}$$

According to Eq. (68) and Eq. (67), the covariance matrix is defined through the interpolation,

$$\Sigma_t = ((1-t)\boldsymbol{I} + t\boldsymbol{T})\boldsymbol{L}_0^{\dagger}((1-t)\boldsymbol{I} + t\boldsymbol{T}) \coloneqq \boldsymbol{R}_t\boldsymbol{L}_0^{\dagger}\boldsymbol{R}_t \tag{71}$$

where $\boldsymbol{R}_t = (1-t)\boldsymbol{I} + t\boldsymbol{T}$. Taking the derivative, we get,

$$\dot{\Sigma}_t = \frac{d}{dt}(\boldsymbol{R}_t\Sigma_0\boldsymbol{R}_t) = \boldsymbol{R}_t'\Sigma_0\boldsymbol{R}_t + \boldsymbol{R}_t\Sigma_0\boldsymbol{R}_t' = (\boldsymbol{T}-\boldsymbol{I})\Sigma_0\boldsymbol{R}_t + \boldsymbol{R}_t\Sigma_0(\boldsymbol{T}-\boldsymbol{I}) \tag{72}$$

Using the fact that $\Sigma_0\boldsymbol{R}_t = \boldsymbol{R}_t\Sigma_0 = \Sigma_t$, we obtain the covariance gradient

$$\dot{\Sigma}_t = (\boldsymbol{T}-\boldsymbol{I})\Sigma_t + \Sigma_t(\boldsymbol{T}-\boldsymbol{I}) \tag{73}$$

So that,

$$\begin{aligned} -\dot{\boldsymbol{L}}_t &= \frac{d(\Sigma_t^{\dagger})}{dt} = \Sigma_t^{\dagger}\frac{d\Sigma_t}{dt}\Sigma_t^{\dagger} = \boldsymbol{L}_t\frac{d\Sigma_t}{dt}\boldsymbol{L}_t \\ &= \boldsymbol{L}_t((\boldsymbol{T}-\boldsymbol{I})\boldsymbol{L}_t^{\dagger} + \boldsymbol{L}_t^{\dagger}(\boldsymbol{T}-\boldsymbol{I}))\boldsymbol{L}_t \\ &= \boldsymbol{L}_t(\boldsymbol{T}-\boldsymbol{I}) + (\boldsymbol{T}-\boldsymbol{I})\boldsymbol{L}_t \\ &= \boldsymbol{L}_t\boldsymbol{T} + \boldsymbol{T}\boldsymbol{L}_t - 2\boldsymbol{L}_t \end{aligned} \tag{74}$$

Thus, $\dot{\boldsymbol{L}}_t = 2\boldsymbol{L}_t - \boldsymbol{L}_t\boldsymbol{T} - \boldsymbol{T}\boldsymbol{L}_t$.

Given that $\boldsymbol{L}_t = \boldsymbol{D}_t - \boldsymbol{W}_t$ so that $\boldsymbol{W}_t = \text{diag}(\boldsymbol{L}_t) - \boldsymbol{L}_t$, taking the derivative gives $\dot{\boldsymbol{W}}_t = \text{diag}(\dot{\boldsymbol{L}}_t) - \dot{\boldsymbol{L}}_t$. As we assume the edges, $E_t \sim \delta(\cdot, \boldsymbol{W}_t)$, the derivate directly yields the velocity,

$$v_t(E_t \mid G_0, G_1) = \dot{\boldsymbol{W}}_t = \text{diag}(\dot{\boldsymbol{L}}_t) - \dot{\boldsymbol{L}}_t.$$

The node feature velocity. The instantaneous velocity field follows,

$$v_t(\mathcal{X} \mid G_0, G_1) = \dot{\boldsymbol{\mu}}_t + \frac{1}{2}\dot{\Sigma}_t\Sigma_t^{-1}\left(\mathcal{X} - \boldsymbol{\mu}_t\right).$$

The mean gradient interpolating η_0 and η_1 can be written as $\dot{\boldsymbol{\mu}}_t = \boldsymbol{X}_1 - \boldsymbol{X}_0$ and $\boldsymbol{X}_t = (1-t)\boldsymbol{X}_0 + t\boldsymbol{X}_1$. So that the velocity leads to,

$$v_t(\mathcal{X} \mid G_0, G_1) = \boldsymbol{X}_1 - \boldsymbol{X}_0 + \frac{1}{2}\dot{\boldsymbol{L}}_t^{\dagger}\boldsymbol{L}_t\left(\mathcal{X} - \boldsymbol{X}_t\right).$$

However, in practice, we do not need such a complicated velocity term. We wish to avoid the estimation of complex gradient-inverse term so that we can escape from the complicated computation. Under the assumption that the amplitude of covariance is much smaller than the mean difference, we can omit the second term and just keep the mean difference. Hence the instantaneous velocity is simply described as

$$v_t(X_t \mid G_0, G_1) = \boldsymbol{X}_1 - \boldsymbol{X}_0 = X_1 - X_0 = \frac{1}{1-t}(\boldsymbol{X}_1 - \boldsymbol{X}_t) \tag{75}$$

D Discrete Bures-Wasserstein flow matching for graph generation

D.1 Probability path construction for discrete Bures-Wasserstein flow matching

The discrete probability path. We design the probability path as discrete distributions,

$$\begin{aligned} &p_t(x_v \mid G_0, G_1) = \text{Categorical}([\boldsymbol{X}_t]_v), \quad p_t(e_{uv} \mid G_0, G_1) = \text{Bernoulli}([\boldsymbol{W}_t]_{uv}) \\ &\text{s.t. } p_0(\mathcal{G}) = \delta(G_0, \cdot), p_1(\mathcal{G}) = \delta(G_1, \cdot) \end{aligned} \tag{76}$$

where $\boldsymbol{W}_t = \boldsymbol{D}_t - \boldsymbol{L}_t$ with $\boldsymbol{X}_t$ and $\boldsymbol{L}_t$ defined the same in Eq. (11). We consider the fact that the Dirac distribution is a special case when the Categorical/Bernoulli distribution has probability 1 or 0, so the boundary condition $p_0(\mathcal{G}) = \delta(G_0, \cdot), p_1(\mathcal{G}) = \delta(G_1, \cdot)$ holds. As such, $\boldsymbol{X}_t = (1-t)\boldsymbol{X}_0 + t\boldsymbol{X}_1 \in [0,1]^{|\mathcal{V}|\times K}$. Since the boundary condition for each entry, $[\boldsymbol{X}_0]_v$ and $[\boldsymbol{X}_1]_v$ are two one-hot embeddings, $[\boldsymbol{X}_t]_v = t[\boldsymbol{X}_0]_v + (1-t)[\boldsymbol{X}_1]_v$ would sum to one, which works as a valid probability vector. Thus, $\text{Categorical}([\boldsymbol{X}_t]_v)$ is a K-class categorical distribution.

For the edge distribution, we just consider e_{uv} is conditionally independent of the other given $[\boldsymbol{W}_t]_{uv}$. One thing to emphasize is that, given the nature of Bures-Wasserstein interpolation, the yielded $\boldsymbol{W}_t$ is not always bounded by $[0,1]$ thus we have to hard-clip the boundary.

D.2 Approximating Wasserstein distance in Bernoulli distributions

To make sure that the individual nodes are structured and developed jointly while doing flow matching, we assume that the $\text{vec}(\mathcal{X})$ still maintains a covariance matrix similar to Eq. (8), which gives $\boldsymbol{\Lambda} = (\nu\boldsymbol{I} + \boldsymbol{L}) \otimes \boldsymbol{V}^\top\boldsymbol{V}$ given that $\mathcal{X}$ is emitted from a latent variable $\mathcal{H}$ through an affine transformation and the latent variable has a covariance matrix $(\nu\boldsymbol{I} + \boldsymbol{L})^{-1}$. Different from the Gaussian case, the latent variable would still be a discrete distribution, so that the affine transformation carries the covariance matrix out.

Unfortunately, the Wasserstein distance between two discrete graph distributions that follow Eq. (13) does not have a closed-form solution given the complex interwined nature. However, it is possible to use the central limit theorem applied to $\mathcal{X}$ so that we can approximate the Wasserstein distance of two Bernoulli distributions with the Gaussian counterpart. This approximation works when we are in high-dimensional case (high dimension means $|\mathcal{V}|d$ is moderately large.), and the OT-distance between two such Bernoulli distributions is well-captured by the corresponding Gaussian formula, which we already introduced in Eq. (57).

With such nature, even though we are not sampling from Gaussian distributions anymore, it is possible to approximate the Wasserstein distance between two multivariate discrete distributions with the Gaussian counterpart, so the conclusions, such as optimal transport displacements, still hold. And we can similarly derive the Bures-Wasserstein velocity as in the next section.

D.3 Velocity for discrete Bures-Wasserstein flow matching

Node velocity. For node-wise, the path of node features $\mathcal{X}_t$ can be re-written as $p_t(\mathcal{X}) = (1-t)\delta(\cdot, \boldsymbol{X}_0) + t\delta(\cdot, \boldsymbol{X}_1)$ so the conditional velocity can be accessed through $v_t(X_t \mid G_0, G_1) = [\delta(\cdot, \boldsymbol{X}_1) - \delta(\cdot, \boldsymbol{X}_t)]/(1-t)$ similar as the derivation in [19].

Edge velocity. For edge-wise, we look into each entry of the adjacency matrix $\boldsymbol{W}$, and consider a time-dependent Bernoulli distribution, the probability density function is:

$$p_t(e_{uv}) = [\boldsymbol{W}_t]_{uv}^{e_{uv}} (1 - [\boldsymbol{W}_t]_{uv})^{1-e_{uv}}, \qquad e_{uv} \in \{0, 1\}. \tag{77}$$

To properly define a velocity $v(x, t)$, it should follow the continuity equation

$$\frac{\partial}{\partial t} p_t(e_{uv}) + \nabla \cdot (p\, v)_t(e_{uv}) \;=\; 0. \tag{78}$$

We use x and y to denote two states of e_{uv} ($p(e_{uv} = x) := p(x), p(e_{uv} = y) := p(y)$), then the divergence term is

$$\nabla \cdot (p\, v)(e_{uv} = x) = \sum_{y \neq x} [p_t(y)\, v_t(y \to x) - p_t(x)\, v_t(x \to y)]. \tag{79}$$

As we are working on a Bernoulli distribution, then the forward equations become

$$\begin{cases} \partial_t p(0) = p(1)\, v_t(1 \to 0) - p(0)\, v_t(0 \to 1), \\ \partial_t p(1) = p(0)\, v_t(0 \to 1) - p(1)\, v_t(1 \to 0). \end{cases} \tag{80}$$

Since $p_t(1) = [\boldsymbol{W}_t]_{uv}$,, we have $\partial_t p(1) = [\dot{\boldsymbol{W}}_t]_{uv}$ and $\partial_t p(0) = -[\dot{\boldsymbol{W}}_t]_{uv}$. Hence

$$p(0)\, v_t(0 \to 1) \;-\; p(1)\, v_t(1 \to 0) = [\dot{\boldsymbol{W}}_t]_{uv}.$$

There are many solutions to the above equation. We chose a symmetric solution so that the transition of $e_{uv} \to 1 - e_{uv}$ with

$$v_t(0 \to 1) = \frac{[\dot{\boldsymbol{W}}_t]_{uv}}{1 - [\boldsymbol{W}_t]_{uv}}, \quad v_t(1 \to 0) = -\frac{[\dot{\boldsymbol{W}}_t]_{uv}}{[\boldsymbol{W}_t]_{uv}}.$$

Finally for concise, we can write write it as a velocity field on states $e_{uv} \in \{0, 1\}$, note $1 - 2e_{uv}$ is $+1$ at $e_{uv} = 0$ and -1 at $e_{uv} = 1$. Thus, we have

$$v(e_{uv}, t) = (1 - 2e_{uv})\, \frac{[\dot{\boldsymbol{W}}_t]_{uv}}{[\boldsymbol{W}_t]_{uv}\, (1 - \boldsymbol{W}_t]_{uv})}, e_{uv} \in \{0, 1\},$$

which in matrix form gives

$$v_t(E_t \mid G_1, G_0) = (1 - 2E_t)\, \frac{\dot{\boldsymbol{W}}_t}{\boldsymbol{W}_t \circ (1 - \boldsymbol{W}_t)}. \tag{81}$$

Combine the node velocity and the edge velocity, we can now introduce the Discrete Bures-Wasserstein Flow matching algorithm, with the training and inference part respectively introduced in Algorithm 3 and Algorithm 4.

E Design space for Bures Wasserstein interpolation and velocity

In the introduction part, we have already compared different probability paths and how they are impacting the inference time sampling. While the Bures-Wasserstein flow path is shown to produce a better probability path for the model to learn, as we illustrated in Fig. 1a, we have to point out that linear interpolation and the corresponding probability path can still converge to the data distribution with sufficiently large flow steps. As if we conduct sampling with infinite flow steps during the later stage of flow, the samples are still able to arrive at the target distributions. A similar pattern exists in diffusion models when they are considered as a Monte-Carlo Markov Chain, and they need sufficiently large steps to converge. We emphasize that the convergence gap in Fig. 1c would be slowly recovered as the number of flow steps increases.

Algorithm 3: Discrete BWFlow Training

Input: Ref. dist p_0 and dataset $\mathcal{D} \sim p_1$.
Output: Trained model $f_\theta(G_t, t)$.

Initialize model $f_\theta(G_t, t)$;
while f_θ *not converged* **do**
  `/* Sample Boundary Graphs */`
  Sample batched $\{G_0\} \sim p_0$, $\{G_1\} \sim \mathcal{D}$;
  `/* Prob.path Construction */`
  Sample $t \sim \mathcal{U}(0, 1)$;
  Calculate the BW interpolation to obtain $\boldsymbol{X}_t, \boldsymbol{W}_t$ via Eq. (11);
  Sample $G_t \sim p(\mathcal{G}_t \mid G_0, G_1)$ according to Eq. (13);
  `/* Denoising - x-prediction */`
  $p^\theta_{1|t}(\cdot \mid G_t) \leftarrow f_\theta(G_t, t)$;
  Loss calculation via Eq. (4);
  optimizer.step();

Algorithm 4: Discrete BWFlow Sampling

Input: Reference distribution p_0, Trained Model $f_\theta(G_t, t)$, Small time step dt,
Output: Generated Graphs $\{\hat{G}_1\}$.

Initialize samples $\{\hat{G}_0\} \sim p_0$;
Initialize the model $p^\theta_{1|t}(\cdot \mid G_t) \leftarrow f_\theta(G_t, t)$
for $t \leftarrow 0$ **to** $1 - dt$ ***by*** dt **do**
  `/* Denoising - x-prediction */`
  Predict $\tilde{G}_1 \leftarrow p^\theta_{1|t}(\cdot \mid G_t)$;
  `/* Velocity calculation */`
  Calculate $v_\theta(\hat{G}_t \mid \hat{G}_0, \tilde{G}_1)$ via Eq. (14);
  `/* Numerical Sampling */`
  Sample $\hat{G}_{t+dt} \sim \hat{G}_t + v_\theta(\hat{G}_t)dt$

Given that different sampling algorithms can all bring the samples to the data distributions under certain conditions, we wish to understand the huge design space of Bures Wasserstein interpolation. We list the advantages and disadvantages in different techniques and discuss further when each techniques should be used.

In general, we consider two important steps to construct the flow matching for graph generation, specifically, *training* and *sampling*. In training, the main challenge is to obtain a valid real velocity $u(G_t)$ to be regressed to, so we listed a few strategies that can help us with that. In sampling, the challenge becomes how to reconstruct the probability path through the velocity estimated.

E.1 The training design

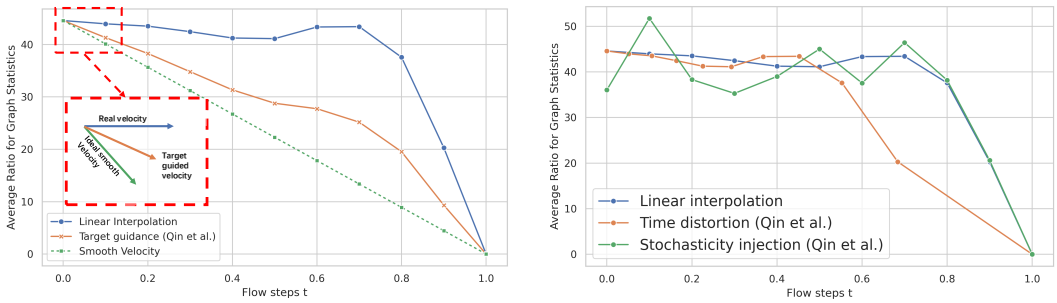
In general, the learning objective in flow matching depends on regressing the velocity term. There are several way to obtain the velocity.

1. **Exact velocity estimation.** Use Eq. (3) as the parameterization and learn $p_\theta(\mathcal{G}_1 \mid G_t)$
2. **Numerical approximation.** In the implementation of [49], the derivative is calculated through numerical approximation. To achieve better efficiency in calculating velocity, we simply consider a numerical estimation as in [49], where the velocity term is obtained as, $\dot{\boldsymbol{L}}_t = (\boldsymbol{L}_{t+\Delta t} - \boldsymbol{L}_t)/\Delta t$. Regressing on the numerical difference can provide an estimation for the velocity.
3. **AutoDiff.** In [11], the derivative of the probability path is evaluated through Pytorch AutoDiff. However, in practice we find this method unstable.

We summarized the training stage model parameterization in Table 4

	Continuous Flow Matching	Discrete Flow Matching
x-prediction	$v^\theta_t(G_t) = \frac{1}{1-t}\left[\tilde{G}^\theta_t(G_t) - G_t\right]$	$v^\theta_t(G_t) = \frac{1}{1-t}\left[p^\theta_{1\|t}(\mathcal{G}_1 \mid G_t) - \delta(\cdot, G_t)\right]$
Numerical Approximation	$v^\theta_t(G_t) \approx G_{t+dt} - G_t$	$v^\theta_t(G_t) \approx p(G_{t+dt} \mid G_0, G_1) - p(G_t \mid G_0, G_1)$
AutoDiff	$v^\theta_t(G_t) \approx \dot{G}_t$	Discrete velocity introduced in Eq. (14)

Table 4: The model parameterization for flow matching in training stage



(a) Linear training path

(b) Training path with manipulation

Figure 4: Techniques for manipulating probability path.

E.2 The sampling design

As we described in the Eq. (3), in our training framework, we actually train a denoised $p_\theta(\mathcal{G}_1 \mid \mathcal{G}_t)$. With such a parameterization and taking discrete flow matching as an example, the sampling can be done through one of the following design choices:

1. **Target-guided velocity sampling.** The velocity is designed as,
$$v_\theta(G_t) = \frac{1}{1-t}(p_\theta(\mathcal{G}_1 \mid G_t) - \delta(G_t, \cdot)).$$
This design directly moves the current point G_t towards the direction pointing to the predicted G_1. The target-guided velocity is guaranteed to converge to the data distribution, but the interpolant might lie outside the valid graph domain.
2. **BW velocity sampling.** We use Eq. (14) to directly estimate the velocity and flow the Bures-Wasserstein probability path to generate new data points. This path is smooth in the sense of graph domain. However, this path requires more computational cost.
3. **Probability path reconstruction.** The third option is directly reconstructing the probability path, i.e., we first obtain an estimated point,
$$\tilde{G}_1 \sim p_\theta(\mathcal{G}_1 \mid G_t)$$
and then construct the data point at $t + dt$, which gives
$$G_{t+dt} \sim p(\mathcal{G}_t \mid \tilde{G}_1, G_0)$$
through Eq. (12). This is the most computationally costly method, which is obtained through the diffusion models. But this method also provide accurate probability path reconstruction.

In Section 4, we show BW velocity follows a path that minimizes the Wasserstein distance thus provides better performance, but sampling following linear velocity also provides convergence with much lower computational cost. So it is a trade-off to be considered in the real-world application.

	Continuous Flow Matching	Discrete Flow Matching
BW Velocity	Eq. (12)	Eq. (14)
Target-guided Velocity	$v_\theta(G_t) = \frac{1}{1-t}(\tilde{G}_1 - G_t)$	$v_\theta(G_t) = \frac{1}{1-t}(p_\theta(\mathcal{G}_1 \mid G_t) - \delta(G_t, \cdot))$
Path Reconstruction	$G_{t+dt} = \delta(G_{t+dt} \mid \tilde{G}_1, G_0)$	$G_{t+dt} \sim p(G_{t+dt} \mid \tilde{G}_1, G_0)$

Table 5: Reconstructing probability path choices in flow matching during inference

F Discussion and limitations

F.1 The implicit manipulation of probability path

Though not explicitly mentioned, Qin et al. [44] makes huge efforts to manipulate the probability path for better velocity estimation by extensively searching the design space, and their finding aligns well

with the statement that the velocity should be smooth and consistently directing to the data points: 1) **Time distortion:** (The oragne line in Fig. 4b) the polynomial distortion of training and sampling focus on the later stage of the probability trajectory, providing better velocity estimation in this area. This uneven sampling strategy is equivalent to pushing the probability path left to make it smooth. 2) **Target guidance:** (The orange line in Fig. 4a) the target guidance directly estimate the direction from a point along the path towards the termination graph, so that the manipulated probability could smoothly pointing to the data distribution. and 3) **Stochasticity injection:** (The green line in Fig. 4a) Stochasticity explores the points aside from the path, which avoid the path to be stuck in the platform area.

F.2 Potential extension to diffusion models

In order to extend the flow matching algorithms with diffusion models, one important thing is to convert the pair-conditioned probability path and velocity to single boundary conditions. For instance, the probability path in flow matching has the form $p(G_t \mid G_1, G_0)$ and the velocity follows $v(G_t \mid G_1, G_0)$. As suggested in [47, 8, 59], the discrete graph diffusion models require a velocity (which is equivalent to a ratio matrix) to perturb the data distribution conditioned on the data points, which we denote as $v(G_t \mid G_1)$. As long as the unilateral conditional velocity has a tractable form, one can first sample a G_1 and get G_t through iteratively doing to:

$$G_{t-dt} = G_t - v(G_t \mid G_1)dt$$

starting from G_1. So that one can easily construct the probability path $p(G_t \mid G_1)$ to fit into the diffusion model framework. In practice, given that we know the explicit form of $v(G_t \mid G_1, G_{t'})$ (just replace G_0 in the expression), the unilateral conditional velocity can be obtained through taking the limitation,

$$v(G_t \mid G_1) = v(G_t \mid G_1, G_t) = \lim_{t' \to t} v(G_t \mid G_1, G_{t'}).$$

Both linear interpolation and our Bures-Wasserstein interpolation can achieve this easily. We just provide a discussion here and will leave this as future work as this paper does not focus on diffusion models but on flow matching models.

F.3 Permutation invariance

The Bures-Wasserstein distance between two graph distributions is not permutation invariant, and the minimal value is obtained through the graph alignment. So ideally, to achieve optimal transport, graph alignment and mini-batch matching could provide a better probability path. However, permutation invariance is not always a desired property since we only want to find a path that better transforms from the reference distribution to the data distributions. As an illustration, the widely used linear interpolation to construct graph flow [44] does not guarantee permutation invariance as well. And it is proved that, if the measurement is based on Wasserstein distance between two Gaussian distributions.

$$\begin{aligned} d_{\mathrm{BW}}(\eta_{\mathcal{G}_0}, \eta_{\mathcal{G}_1}) &\le d_{\text{Arithmetic}}(\eta_{\mathcal{G}_0}, \eta_{\mathcal{G}_1}) \\ \text{with } d_{\mathrm{BW}}(\eta_{\mathcal{G}_0}, \eta_{\mathcal{G}_1}) &= \|\boldsymbol{X}_0 - \boldsymbol{X}_1\|_F^2 + \beta \operatorname{trace}\left(\boldsymbol{L}_0^\dagger + \boldsymbol{L}_1^\dagger - 2\left(\boldsymbol{L}_0^{\dagger/2}(\boldsymbol{P}^\top \boldsymbol{L}_1 \boldsymbol{P})^\dagger \boldsymbol{L}_0^{\dagger/2}\right)^{1/2}\right), \\ \text{and } d_{\text{Arithmetic}}(\eta_{\mathcal{G}_0}, \eta_{\mathcal{G}_1}) &= \|\boldsymbol{X}_0 - \boldsymbol{X}_1\|_F^2 + \beta \|\boldsymbol{L}_0 - \boldsymbol{P}^\top \boldsymbol{L}_1 \boldsymbol{P}\|^2, \forall \boldsymbol{P} \in \text{ potential permutation set} \end{aligned} \tag{82}$$

G Related works

G.1 Diffusion and flow Models

Among contemporary generative models, diffusion [24] and flow models [33] have emerged as two compelling approaches for their superior performance in generating text and images. In particular, these generative models can be unified under the framework of stochastic interpolation [1], which consists of four procedures [34] as we introduced in Section 1. These contemporary generative models rely on constructing a probability path between data points of an easy-to-sample reference distribution and of the data distribution, and training a machine learning model to simulate the process [34]. So that one can sample from the reference (a.k.a source) distribution and iteratively

transform it to approximate data samples from the target distribution. Diffusion models construct the probability path with a unilateral path conditioned on the data distribution, where one start sampling from a data point X_1 and construct the path $p(\mathcal{X}_t \mid X_1)$. While flow models can condition on either two boundary conditions, $\{X_1, X_0\}$ or just one-side boundary condition X_1.

Depending on the space that the algorithm operates on, both models can be categorized into continuous or discrete models. The continuous generative models assume the data distributions are themself lying in continuous space (such as Gaussian) and build models, with examples in diffusion [24, 48, 57] and flow [33, 36]. The discrete generateive models assume the data follows a discrete distribution, for instance categorical or Bernoulli distributions. Examples include discrete diffusion [8, 50] and discrete flow models [9, 19, 39].

Under the stochastic interpolation framework, the interpolation methods are commonly selected through optimal transport (OT) displacement interpolant [36, 1, 38]. Optimal transport is a classical topic in mathematics that was originally used in economics and operations research [56], and has now become a popular tool in generative models. OT aims for finding the best transport plan between two probability measures with the smallest associated transportation cost. It has been shown that generative models can be combined with technologies such as iterative matching [53] and mini batching [43] to approximate the OT cost, and get a significant boost in their performance in generative modeling.

G.2 Graph generation models

Thanks to the capability of graphs in representing complex relationships, graph generation [64, 35] has become an essential task in various fields such as protein design [26], drug discovery [6], and social network analysis [31]. The initial attempt at graph generation is formalized through autoregression. For instance, GraphRNN [61] organizes the node interactions into a series of connection events and conducts autoregressive prediction for generation. Later, one shot generation methods such as Variational Graph Auto-Encoder were proposed [30, 10].

Among various generative models, diffusion models and flow-based models have emerged as two compelling approaches for their ability to achieve state-of-the-art performance in graph generation tasks [40, 54, 18, 44, 25]. In the early stage, continuous diffusion models were first extended to the task of graph generation [40], where they just view the adjacency matrix as a special signal living on the $\mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ domain. However, these methods fail to capture the natural discreteness of graphs, and Vignac et al. [54] first brings discrete diffusion into graph generation. After that, more work [47, 59] starts to focus on designing better discrete diffusion models for graph generation.

On the other hands, with the development of flow matching techniques, a few works have been developed to utilize flow models for graph generation and they have achieved huge success. Eijkelboom et al. [18] utilizes variational flow matching to process categorical data and Qin et al. [44] developed discrete flow matching for graph generation tasks.

In parallel, there are a number of work that have managed to respect the intrinsic nature of graphs, such as global patterns. For instance, Jo et al. [28] brings a mixture of graph technique to enhance the performance by explicitly learning final graph structures; Yu and Zhan [62] mitigates exposure bias and reverse-start bias in graph generation; Hou et al. [25] improves graph geneartion through optimal transport flow matching techniques but they still assume the independence between nodes and edges and use hamming distance to measure the transport cost; and Li et al. [31] gives the large-scale attributed graph generation framework through batching edges.

However, there remain a core challenge: constructing the probability path p_t. Existing text and image generative models, operating either in the continuous [24, 48, 33, 36] or discrete [8, 50, 9, 19, 39] space, typically rely on linear interpolation between source and target distributions to construct the path. Graph generation models, including diffusion [40, 54, 23, 59, 47] and flow-based models [18, 44, 25], inherit this design by modeling every single node and edge independently and linearly build paths in the disjoint space. However, this approach is inefficient because it neglects the strong interactions and relational structure inherent in graphs, i.e., the significance of a node heavily depends on the configuration of its neighbors. While empirical success have been achieved via fine-grained searching on the training and sampling design [44] such as target guidance and time distortion, we argue that there remains a fundamental issue of the linear probability path construction, and these strategies only mitigate the problem by manipulating the probability path.

Table 6: Training and sampling time on each dataset. TG means using target-guided velocity; BW means using BW velocity.

Dataset	Min Nodes	Max Nodes	Training Time (h)	Graphs Sampled	Sampling Time (h)
Planar	64	64	45 (1.55x)	40	0.07(TG); 0.13 (BW)
Tree	64	64	10 (1.25x)	40	0.07(TG);0.14(BW)
SBM	44	187	74 (0.98x)	40	0.07(TG) 0.14(BW)
Moses	8	27	35 (0.76x)	25000	5(TG); 6(BW)
Guacamol	2	88	251 (1.8x)	10000	7(TG); 21(BW)
QM9	3	29	15	25000	5(TG) 6(BW)
GEOM	/	181	141	10000	7(TG) 14(BW)

Table 7: Best Configuration for Training and Sampling when producing Tables 1 to 3.

	Train		Sampling	
Dataset	Initial Distribution	Train Distortion	Sample Distortion	Sampling steps
Planar	Marginal	Identity	Identity	1000
Tree	Marginal	Polydec	Polydec	1000
SBM	Absorbing	Identity	Identity	1000
MOSES	Marginal	Identity	Identity	500
GUACAMOL	Marginal	Identity	Identity	500
QM9	Marginal	Identity	Identity	500
GEOM	Marginal	Identity	Identity	500

H Comparison with other interpolation methods

In the experimental part, we compare our methods with arithmetic (linear) interpolation, geometric interpolation and harmonic interpolation. We state the equation of them respectively as follows.

We consider the boundary graph G_0 and G_1 with $\boldsymbol{X}_0, \boldsymbol{X}_1 \in \mathbb{R}^{|\mathcal{V}|\times d}$ and $\boldsymbol{W}_0, \boldsymbol{W}_1 \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$. Let $t \in [0,1]$, we fixed the feature interpolation as,

$$\boldsymbol{X}_t = (1-t)\,\boldsymbol{X}_0 \;+\; t\,\boldsymbol{X}_1,$$

the graph structure interpolation can be expressed as,

Linear interpolation:

$$\boldsymbol{W}_t = (1-t)\,\boldsymbol{W}_0 \;+\; t\,\boldsymbol{W}_1.$$

Geometric interpolation:

$$\boldsymbol{W}_t = \boldsymbol{W}_0^{1/2}\left(\boldsymbol{W}_0^{-1/2}\,\boldsymbol{W}_1\,\boldsymbol{W}_0^{-1/2}\right)^t \boldsymbol{W}_0^{1/2},$$

Harmonic interpolation:

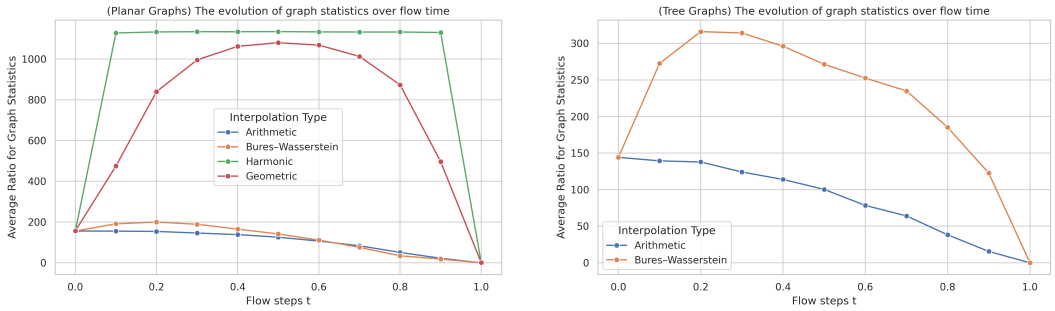
$$\boldsymbol{W}_t = \left((1-t)\,\boldsymbol{W}_0^{-1} \;+\; t\,\boldsymbol{W}_1^{-1}\right)^{-1}.$$

Each interpolation methods actually handle each special manifold assumption, which should be designed under a comprehensive understanding of the task. In our experimental part, we conduct intensive analysis on the impact of interpolation methods to the graph generation quality.

I Additional experiment results

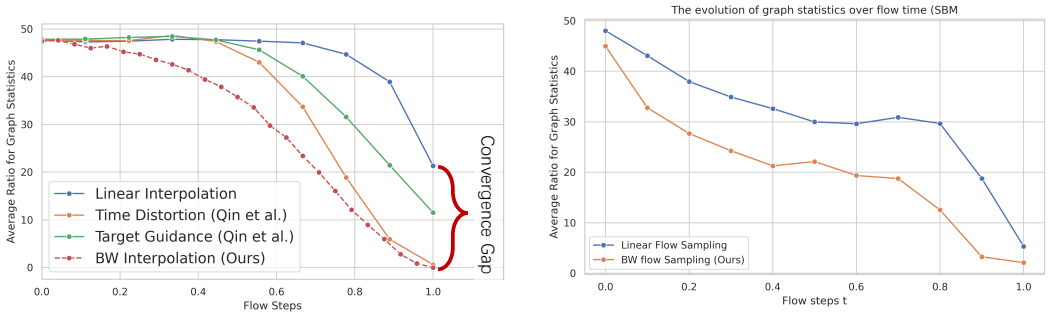
I.1 Experiment setups and computational cost

The training and sampling computation time are provided in Table 6. The experiments were run on a single NVIDIA A100-SXM4-80GB GPU. The hyperparameter configuration in producing Tables 1 to 3 is reported in Table 7.



(a) Probability Path for Planar Graphs (b) Tree Graph Probability Path

Figure 5: BW probability paths for planar and tree graphs.



(a) Sampling path comparision on Planar dataset (b) Sampling path comparision on SBM dataset

Figure 6: The probability path reconstruction in the sampling stage on a) Planar graphs and b) SBM graphs.

I.2 Additional results for the training paths

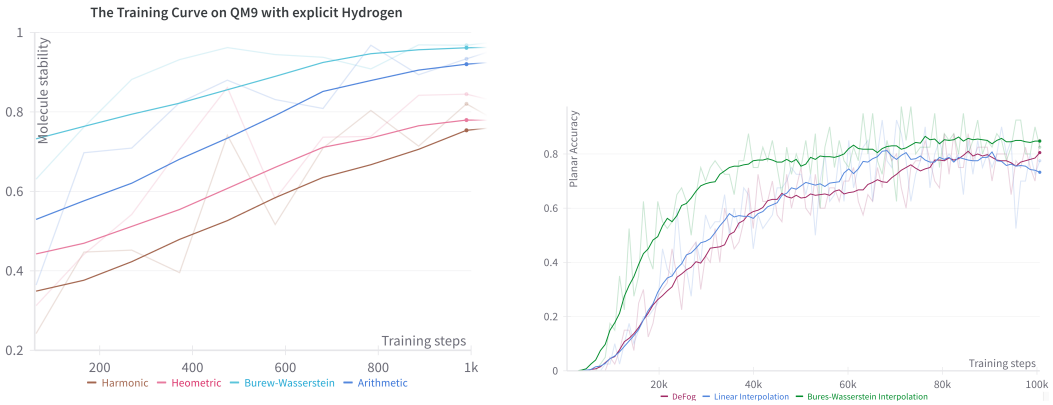
Fig. 5 gives the training probability path construction for planar graphs and tree graphs. While planar graphs have a similar pattern as the SBM datasets as in Fig. 3a, the probability path constructed for tree graphs does not follow a similar pattern. We attribute this to the different geometry of tree graphs that reside in hyperbolic space [60].

I.3 More experiments on plain graph generations

Additional results for sampling paths. We then give the sampling path construction in Fig. 6. To better illustrate the advantage of BWFlow, we fix the sampling steps to be as small as 50. It is clear that in planar and SBM dataset, the BW velocity can still provide a smooth probability and stable convergence towards the data distribution. While the linear velocity does not give a good probability path and fails to converge to the optimal value, especially when the sampling size is small.

The maximum mean discrepancy (MMD) of four graph statistics between the set of generated graphs and the test set is measured, including degree (Deg.), clustering coefficient (Clus.), count of orbits with 4 nodes (Orbit), the eigenvalues of the graph Laplacian (Spec.), wavelet ratio (Wavelet.). To verify that the model learns to generate graphs with valid topology, we gives the percentage of valid, unique, and novel (V.U.N.) graphs for where a valid graph satisfies the corresponding property of each dataset (Planar, Tree, SBM, etc.).

Full results for plain graph generation. Table 8 gives the full results with other generative models aside from the diffusion and flow models. Table 9 gives the results on smaller datasets, i.e., comm20



(a) QM9 with explicit hydrogen (b) Planar

Figure 7: Training curves on QM9 and planar datasets with explicit hydrogen.

I.4 Full table for the synthetic graph generation

In Table 10, we illustrate the numerical results for comparing the interpolation methods in plain graph generation without node features. It is clear that BWFlow outperforms other methods in planar and SBM graphs. But the performance was not good in tree graph generations.

I.5 3D Molecule Generation: QM9 without explicit hydrogen

In Table 11 we report the results of QM9 without explicit hydrogen. This task is relatively easy compared to the generation task with explicit hydrogen, and both Midi and our BWFlow have achieved near-saturated performance with validity near to 100%.

I.6 Convergence Analysis

Fig. 7 are the training convergence analysis on Planar and QM9 dataset, showing that BWFlow provides a fast convergence speed than others.

Table 8: Graph Generation Performance on Synthetic Graphs. Results are obtained through tuning the probability path manipulation techniques. The remaining values are obtained from Qin et al. [44].

	Planar Dataset				
Model	Ratio ↓	Valid ↑	Unique ↑	Novel ↑	V.U.N. ↑
Train set	1.0	100	100	0.0	0.0
GRAN [32]	2.0	97.5	85.0	2.5	0.0
SPECTRE [37]	3.0	25.0	100	100	25.0
DiGress [54]	5.1	77.5	100	100	77.5
EDGE [13]	431.4	0.0	100	100	0.0
BwR [15]	251.9	0.0	100	100	0.0
BiGG [14]	16.0	62.5	85.0	42.5	5.0
GraphGen [20]	210.3	7.5	100	100	7.5
HSpectre (one-shot) [4]	1.7	67.5	100	100	67.5
HSpectre [4]	2.1	95.0	100	100	95.0
GruM [28] 1.8	—	—	—	90.0	
CatFlow [18]	—	—	—	—	80.0
DisCo [59]	—	83.6 ±2.1	100.0 ±0.0	100.0 ±0.0	83.6 ±2.1
Cometh - PC [47]	—	99.5 ±0.9	100.0 ±0.0	100.0 ±0.0	**99.5** ±0.9
DeFoG	1.6 ±0.4	99.5 ±1.0	100.0 ±0.0	100.0 ±0.0	**99.5** ±1.0
BWFlow	**1.3** ±0.4	97.5 ±2.5	100.0 ±0.0	100.0 ±0.0	97.5±2.5

	Tree Dataset				
Train set	1.0	100	100	0.0	0.0
GRAN [32]	607.0	0.0	100	100	0.0
DiGress [54]	**1.6**	90.0	100	100	90.0
EDGE [13]	850.7	0.0	7.5	100	0.0
BwR [15]	11.4	0.0	100	100	0.0
BiGG [14]	5.2	100	87.5	50.0	75.0
GraphGen [20]	33.2	95.0	100	100	95.0
HSpectre (one-shot) [4]	2.1	82.5	100	100	82.5
HSpectre [4]	4.0	100	100	100	**100**
DeFoG	1.6 ±0.4	96.5 ±2.6	100.0 ±0.0	100.0 ±0.0	96.5 ±2.6
BWFlow	**1.4**±0.3	75.5 ±2.4	100.0±0	100.0±0	75.5±2.4

	Stochastic Block Model ($n_{\max}$ = 187, n_{avg} = 104)				
Model	Ratio ↓	Valid ↑	Unique ↑	Novel ↑	V.U.N. ↑
Training set	1.0	85.9	100	0.0	0.0
GraphRNN [61]	14.7	5.0	100	100	5.0
GRAN [32]	9.7	25.0	100	100	25.0
SPECTRE [37]	2.2	52.5	100	100	52.5
DiGress [54]	1.7	60.0	100	100	60.0
EDGE [13]	51.4	0.0	100	100	0.0
BwR [15]	38.6	7.5	100	100	7.5
BiGG [14]	11.9	10.0	100	100	10.0
GraphGen [20]	48.8	5.0	100	100	5.0
HSpectre (one-shot) [4]	10.5	75.0	100	100	75.0
HSpectre [4]	10.2	45.0	100	100	45.0
GruM [28]	**1.1**	—	—	—	85.0
CatFlow [18]	—	—	—	—	85.0
DisCo [59]	—	66.2 ±1.4	100.0 ±0.0	100.0 ±0.0	66.2 ±1.4
Cometh [47]	—	75.0 ±3.7	100.0 ±0.0	100.0 ±0.0	75.0 ±3.7
DeFoG	4.9 ±1.3	90.0 ±5.1	100.0 ±0.0	100.0 ±0.0	90.0 ±5.1
BWFlow	3.8±0.9	**90.5** ±4.0	100.0 ±0.0	100.0 ±0.0	**90.5** ±4.0

FM type	Interpolation	comm-20		
		Deg.	Clus.	Orbit.
Discrete	Arithmetic	0.071	0.115	0.037
	Harmonic	0.011	0.036	0.027
	Geometric	0.047	0.083	0.02
	BW	0.009	0.013	0.017

Table 9: Quantitative experimental results on COMM20 (smaller dataset).

Dataset	FM type	Interpolation	V.U.N metrics			Spectral Metrics					
			Novelty	Uniqueness	Validity	Orbit	Spec	Clustering	Degree	Wavelet	Avg. Ratio
Planar	Discrete	DeFog	100	100	78.25	8.98	1.45	2.09	2.65	2.38	3.51
		Arithmetic	100	100	73.25	10.83	1.33	1.74	2.24	2.39	3.70
		harmonic	100	100	0.00	4519.63	2.57	17.01	25.10	42.41	921.35
		Geometric	100	100	23.25	655.66	1.61	10.17	13.25	6.68	137.47
		BW	100	100	84.75	5.14	1.27	1.69	1.78	2.02	2.38
Tree	Discrete	Defog	100	100	61.53	/	1.17	/	1.27	1.51	1.32
		Arithmetic	100	100	56.91	/	1.16	/	1.04	1.45	1.22
		harmonic	100	100	0.53	/	2.32	/	1.93	3.31	2.52
		Geometric	100	100	48.38	/	1.62	/	2.13	2.10	1.94
		BW	100	100	51.45	/	1.58	/	2.56	2.13	2.09
SBM	Discrete	Arithmetic	100	100	44.63	2.57	1.40	1.46	15.55	7.88	5.77
		harmonic	100	100	9.73	3.10	10.23	1.59	172.10	103.04	58.10
		Geometric	100	100	0.0	3.11	4.45	1.80	150.41	60.60	54.65
		BW	100	100	58.70	2.03	1.50	1.50	9.04	8.41	4.51

Table 10: Ablation study on interpolation methods when probability path manipulation techniques are all disabled. The clustering and orbit ratios in tree graphs are omitted, given that in the training set, the corresponding statistics are 0. The results go over exponential moving average (decay 0.999) for the last 5 checkpoints. The table is produced with Marginal boundary distributions, without time distortion.

Dataset	Interpolation	Metrics					
		μ	V.U.N(%)	Connected(%)	Charges(10^{-2})	Atom(10^{-2})	Angles(°)
QM9 (w/o h)	MiDi	1.00	98.0	100.0	0.4	5.1	1.49
	Linear Flow	1.60	79.33	52.3	0.7	14.0	8.77
	BWFlow	1.02	99.8	100.0	0.4	4.8	1.53

Table 11: Quantitative experimental results on QM9 datasets without explicit hydrogen in 3D molecule generation.

Table 12: Graph generation performance on the synthetic datasets: Planar, Tree, and SBM. Given that the synthetic datasets are usually unstable in evaluation, we applied an exponential moving average to stabilize the results and sample 5 times (each run generates 40 graphs) to calculate the mean and standard deviation. The experiment settings are in Table 7.

		Planar		Tree		SBM	
Model	Class	V.U.N. ↑	A.Ratio ↓	V.U.N. ↑	A.Ratio ↓	V.U.N. ↑	A.Ratio ↓
Train set	—	100	1.0	100	1.0	85.9	1.0
DiGress [54]	Diffusion	77.5	5.1	90.0	**1.6**	60.0	1.7
EDGE [13]	Diffusion	0.0	431.4	0.0	850.7	0.0	51.4
HSpectre [4]	Diffusion	95.0	2.1	**100.0**	4.0	75.0	10.5
GruM [28]	Diffusion	90.0	1.8	—	—	85.0	**1.1**
CatFlow [18]	Flow	80.0	—	—	—	85.0	—
DisCo [59]	Diffusion	83.6±2.1	—	—	—	66.2±1.4	—
Cometh [47]	Diffusion	**99.5**±0.9	—	—	—	75.0±3.7	—
DeFoG [44]	Flow	**99.5**±1.0	1.6±0.4	96.5±2.6	1.6±0.4	90.0±5.1	4.9±1.3
BWFlow	Flow	97.5±2.5	**1.3**±0.4	75.5±2.4	1.4±0.3	**90.5**±4.0	3.8±0.9
DeFoG (EMA)	Flow	77.5±8.37	3.5±1.7	**73.1**±11.4	1.50±0.3	**85.0**±7.1	3.7±0.9
BWFlow (EMA)	Flow	**84.8**±6.44	**2.4**±0.9	68.1±12.2	**1.24**±0.2	83.5±6.0	**2.4**±0.6

Table 13: Comparison of interpolation methods on 3D Molecule generation with explicit hydrogen in QM9 dataset.

Flow Type	Interpolation	Metrics							
		μ	V.U.N(%)	Mol Stable	Atom Stable	Connected(%)	Charge(10^{-2})	Atom(10^{-2})	Angles(°)
Discrete	MiDi	1.01	93.13	93.98	99.60	99.21	0.2	3.7	2.21
	Arithmetic	1.01	87.53	88.45	99.13	99.09	0.4	4.2	2.72
	harmonic	1.01	94.91	94.54	99.65	99.03	0.6	6.4	2.21
	Geometric	1.01	91.26	91.29	99.42	98.42	0.1	4.4	3.63
	BW	1.01	**96.45**	**97.84**	**99.84**	99.24	**0.1**	**2.3**	**1.96**
Continuous	Arith	2.15	25.45	10.23	76.85	28.82	0.7	**5.6**	14.47
	harmonic	1.01	11.38	11.64	73.48	99.65	1.2	17.2	15.04
	Geometric	**1.00**	42.07	46.08	91.13	**99.87**	1.0	12.7	8.03
	BW	1.02	**62.02**	**61.76**	**95.99**	97.72	**0.6**	8.7	**7.80**

* Clearly, continuous flow matching models are not as comparative as discrete flow matching models.

Table 14: Large molecule generation results. Only iterative denoising-based methods are reported.

	Guacamol					MOSES						
Model	Val. ↑	V.U. ↑	V.U.N.↑	KL div ↑	FCD ↑	Val. ↑	Unique. ↑	Novelty ↑	Filters ↑	FCD ↓	SNN ↑	Scaf ↑
Training set	100.0	100.0	0.0	99.9	92.8	100.0	100.0	0.0	100.0	0.01	0.64	99.1
DiGress [54]	85.2	85.2	85.1	92.9	68.0	85.7	**100.0**	95.0	97.1	**1.19**	0.52	14.8
DisCo [59]	86.6	86.6	86.5	92.6	59.7	88.3	**100.0**	**97.7**	95.6	1.44	0.50	15.1
Cometh [47]	98.9	98.9	97.6	96.7	72.7	90.5	99.9	92.6	**99.1**	1.27	0.54	16.0
DeFoG [44]	**99.0**	**99.0**	**97.9**	**97.7**	**73.8**	**92.8**	99.9	92.1	98.9	1.95	**0.55**	14.4
BWFlow (Ours)	98.8	98.9	97.4	/	69.2	92.0	**100.0**	94.5	98.4	1.32	**0.56**	15.3

Table 15: Large molecule generation results. Only comparing the representative diffusion and flow models. B.E. is the scenario that only considers binary edge types. The results are almost saturated, thus not very informative.

	Guacamol			MOSES		
Model	Val. ↑	V.U. ↑	V.U.N.↑	Val. ↑	Unique. ↑	Novelty ↑
Digress (B.E.)	96.0	98.9	97.4	96.1	100	100
Defog (B.E.)	98.4	98.4	97.9	99.3	100	100
BWFlow (B.E.)	98.0	98.0	97.7	99.6	100	100