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, particularly diffusion and flow-based models, have achieved superior generative performance through constructing an iterative transformation from a simple reference distribution to the data distribution. However, these methods typically use linear interpolations to build the path grounded in Euclidean assumptions, which do not hold for graphs due to their intrinsic non-Euclidean structure. To address this gap, we represent graphs as statistical objects parameterized by Markov random fields (MRF), which permits a closed-form solution for the sampling distribution, transportation distance, and inherent interpolations. Building upon these results, we leverage the corresponding optimal transport displacement interpolation under the MRF assumption and revisit the probability path construction for graph generation. Following the probability path constructed, we then introduce Bures-Wasserstein Flow Matching(BWFlow), which integrates the results into a flow-matching model tailored for graph generation. The novel framework can be adapted to both continuous and discrete flow-matching frameworks depending on the task. Experimental evaluations in plain graph generation and 2D/3D molecule generation validate the effectiveness of BWFlow in preserving key statistical properties while accurately modeling the intricate relationships in graph-structured data.

1 Introduction

Due to graphs' capacity to represent complex relationships, graph generation [66, 35] has become an essential task in various fields such as protein design [25], drug discovery [7], and social network analysis [32]. 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 [42, 57, 16, 48, 24]. In particular, these contemporary generative models rely on constructing a probability path that transforms from the data distribution to an easy-to-sample reference distribution and training a machine learning model to revert the process back [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.

The flow-based and diffusion models can be unified under the framework of stochastic interpolation [1], which consists of four procedures. 1) Drawing samples from the data distribution $p_1(\cdot)$ and/or reference distribution $p_0(\cdot)$; 2) Constructing a time-continuous probability path $p_t(\cdot), 0 \leq t \leq 1$ interpolating between p_0 and p_1; 3) Train a model to reconstruct the probability path; and 4) sampling from p_0 and transforming it through the model trained to get samples that approximately follow p_1.

Within such a framework, an important component is constructing the probability path p_t. Existing methods widely consider a linear interpolation between source and target distributions to construct the

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

Preprint. Under review.

probability path, which is derived through the optimal transport (OT) displacement interpolant [56]. This includes continuous generative methods with diffusion [23, 51, 60] and flow [33, 37], as well as the discrete generative models like discrete diffusion [9, 53] and discrete flow [10, 18, 40]. However, such paths have been designed principally under the assumption that the data points lie in the Euclidean space and the cost function for Wasserstein distance is measured by the L^2 norm. Existing models for graph generation, including the diffusion-based models [42, 57, 21, 63, 50] and the flow-based ones [16, 48, 24], widely utilized "straight" probability path in the development of their models. However, graphs are objects lying in a non-euclidean space given their irregular structures and relationships. Such property violates the Euclidean and isotropic assumptions that works as the fundamental building block for linear interpolation. As such, extending diffusion/flow models to graph generation still presents unique challenges.

This observation raises the necessity to re-design the stochastic interpolation methods for graph generative models. To this end, we name two crucial research questions (RQs) when adapting the four-step framework to graph generations, specifically,

1. *(RQ1): How to view graphs as statistical objects, so that one can extend the stochastic interpolation framework between two distributions for graph generations? This includes the definition of distributions, the transportation distances, and the interpolation methods following OT displacement.*

2. *(RQ2): How could we develop the flow-matching model for graph generation based on the derived OT interpolation? This includes the construction of the probability path, the parameterization of the velocity field, the design of training objective and the inference sampling algorithms.*

To solve the first challenge (RQ1), we borrow the idea from statistical relational learning and consider Markov Random Fields (MRF) as a way to model the graph distributions. MRF was initially utilized in statistical physics to model the dynamics of an interconnected multi-body system, such as molecules and proteins [62, 3]. Similarly, any graph can be viewed as a connected network and the joint evolution dynamics of node features and graph structure can be captured by MRF models. Through MRF, the graphs are modeled by a colored Gaussian distribution whose covariance matrix are uniquely determined by the graph structure. Extending from [20], we further derive the Wasserstein distance between two statistical graph objects. The Wasserstein distance is then utilized to construct the Bures-Wasserstein interpolation of two graphs, which works as a replacement of the straight interpolation that does not guarantee the optimal transport displacement in graph generations

To answer RQ2, the results are further used in the development of flow matching for graph generation. Specifically, we utilize the Bures-Wasserstein interpolation to construct the probability path, which thereby models the co-evolution of node features and graph structure and ensures the interpolated points preserve the graphs' statistical properties by limiting the domain of interpolant. We show that the designed method can be plugged into both continuous and discrete flow frameworks to develop generative models for graph-structured data, which we name Bures-Wasserstein Flow (BWFlow).

To highlight the versatility of our BWFlow method, we test on two tasks: the Discrete Graph Generation and the De-novo Molecule Generation. While existing flow-matching models have already achieved near-saturated performance in the discrete graph generation task, our model inherits the advantage and excess in the task in the general performance. In de-novo molecule generation, our model outperforms the existing ones by a margin. To further illustrate the effectiveness of flowing through the Bures-Wasserstein interpolation, we conduct ablation studies on the types of interpolations within the framework of both continuous and discrete flow matching. We observe that our BWFlow consistently outperforms the competitors - the flow-matching models constructed through arithmetic, geometric, and harmonic interpolations. Finally, we conducted the convergence analysis and found that BWFlow grants better convergence rate and

2 Preliminaries

2.1 Flow Matching and Conditional Flow Matching

Flow Matching (FM). We consider the setting of a pair of data distributions over state space $\mathcal{S}$ with densities $q_0(\mathcal{X})$ and $q_1(\mathcal{X})$[2]. Generative modeling considers the task of fitting a mapping

[2]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 distribution parameters, i.e. $X \sim p(\mathcal{X} \mid \boldsymbol{X})$.

from $\mathcal{S}$ to $\mathcal{S}$ that transforms $X_0 \sim q_0$ to $X_1 \sim q_1$. Continuous Normalizing Flow (CNF) [12] parameterizes the transformation through a push-forward equation that interpolates between q_0 and q_1 and constructs a probability path $p_t(\mathcal{X}) = [\psi_t{}_\sharp q_0](\mathcal{X})$ through a time-dependent function ψ_t (a.k.a flow). A vector field u_t is said to generate p_t between the two distributions if its flow ψ_t satisfies, $\frac{d}{dt}\psi_t(\mathcal{X}) = u_t(\psi_t(\mathcal{X})), \psi_0(\mathcal{X}) = \mathcal{X}$. For simplicity we denote $u_t(\psi_t(\mathcal{X})) := u_t(\mathcal{X})$. The FM objective [33] is designed to match the real velocity field, which yields the training objective:

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

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

Conditional Flow Matching (CFM). Given that the actual vector field and the path are not tractable [56], 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 probability path and the velocity field are tractable. As an example, linear CFM introduced in [56] set $\pi(\cdot) = q_0(\mathcal{X})\, q_1(\mathcal{X})$ to be the independent coupling. Then the conditional flows has a closed form,

$$p_t(\mathcal{X} \mid Z) = \mathcal{N}\left(\mathcal{X} \mid tX_1 + (1-t)X_0, \sigma^2\right) \text{ and } u_t(\mathcal{X} \mid X_0, X_1) = X_1 - X_0. \tag{2}$$

The CFM aims at regressing a velocity field $v_\theta(\cdot)$ to $u_t(\mathcal{X} \mid Z)$, which defines the training objective:

$$\mathcal{L}_{\mathrm{CFM}}(\theta) := \mathbb{E}_{t, Z\sim\pi(\cdot), p_t(\cdot\mid Z)} \left\| v_\theta(X_t) - u_t(X_t \mid Z) \right\|^2, \tag{3}$$

where it is shown that the CFM optimization has the same optimum as the FM objective [56].

2.2 Optimal Transport and Flow Matching

A natural question is the design of conditional probability path $p_t(\mathcal{X} \mid Z)$. Existing literature [37, 1] argues that the path should be chosen to recover the optimal transport (OT) displacement interpolant [39]. OT is a classical topic in mathematics that was originally used in economics and operations research [59], and has now become a popular tool in generative models. Specifically, 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)**.** Let (M, d) be a metric space, and let η_0 and η_1 be two probability measures on M. We denote the possible coupling space for η_0 and η_1 as $\Pi(\eta_0, \eta_1)$. A coupling $\pi \in \Pi(\eta_0, \eta_1)$ is a joint probability measure on $M \times M$ whose marginals are η_0 and η_1 respectively. With $c(X, Y)$ being the cost of transporting the mass between X and Y (e.g. the squared difference), the Wasserstein distance is defined as,

$$(W_c(\eta_0, \eta_1))^2 = \inf_{\pi \in \Pi(\eta_0,\eta_1)} \int_{M\times M} c(X, Y)\, d\pi(X, Y). \tag{4}$$

Specifically, the probability path as shown in Eq. (2) with $\sigma^2 \to 0$ is a solution to Eq. (4) when q_0 is the standard Gaussian distribution and q_1 can be approximated by a Gaussian distribution. Combined with technologies such as iterative matching [56] and mini batching [47] to approximate the optimal transport cost in Eq. (4), CFM has become the leading method in generative modeling.

However, recent research in manifold flow matching [11, 29] suggests that the interpolation trajectories are not appropriate if $\mathcal{X}$ is not designed under the assumption of Euclidean geometry. This calls for the necessity of revisiting the trajectory in generating graphs, a non-euclidean object, as well.

3 Methodology

While the straight flow is proven to be the optimal transport interpolation considering the Euclidean geometry when the data in $\mathbb{R}^d$, the graph lies in a non-euclidean space thus the linear interpolation of the graph structure and node features does not simply grant optimal transport cost in such a space. In this section, we first consider graphs as a statistical object parameterized by Markov Random Fields in Section 3.2 to solve RQ1. We then develop the relevant OT distance and interpolation methods in Section 3.3. We further develop the flow matching for graphs (BWFlow) in Section 3.4 to solve RQ2. Finally, we extend our method to both continuous and discrete flow matching for more general graph generation tasks in Section 3.5.

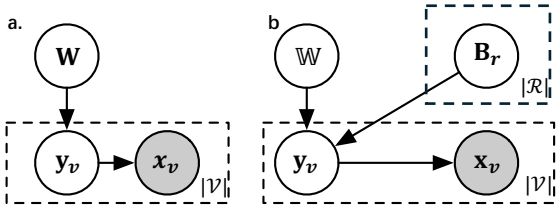


Figure 1: The graphical models for a) HMN and b) our H2MN. The shadowed variable is observable. Give a visualization of Markov Random Field for molecule dynamics

3.1 Conditional Flow Matching for Graph Generation

Graphs as statistical objects. CFM is defined over a trajectory between p_1 and p_0. When considering graph generation, the very first step is to define a data-generating process and model graphs as a statistical object. For notation, we let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{X}\}$ denote an undirected graph random variable with edges $\mathcal{E}$, nodes $\mathcal{V}$, and node features $\mathcal{X}$. 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 matrices $\boldsymbol{W} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$, Laplacian matrix as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$, with $\boldsymbol{D} = \operatorname{diag}(\boldsymbol{W}\boldsymbol{1})$ being the degree matrix. In a nutshell, graphs are statistical objects sampled from $G \sim p(\mathcal{G} \mid \boldsymbol{G}) = p(\mathcal{X}, \mathcal{E} \mid \boldsymbol{X}, \boldsymbol{W})$.

A common selection to decompose the probability density assumes the node features and graph structure are independent [24, 48, 16], which gives $p(\mathcal{G}) = p(\mathcal{X}) \cdot p(\mathcal{E})$. Then, the boundary conditions follow Dirac distributions such that $p_i(\mathcal{G}) = \delta(\mathcal{X}_i = \boldsymbol{X}_i) \cdot \delta(\mathcal{E}_i = \boldsymbol{W}_i), \forall i = \{0, 1\}$. Such a decomposition causes reverse-starting and exposure biases in graph generation tasks [64]. Unlike them, we aim to mitigate the biases and construct an optimal transport trajectory through modeling the interdependency between node features and graph structures, as we will introduce later.

Graph CFM. We now consider constructing a probability path for graph generation that satisfies $p_t(\mathcal{G}_t) = \mathbb{E}_{(G_0, G_1) \sim \pi(\cdot,\cdot)} p_t(\mathcal{G}_t \mid G_0, G_1)$. While our method is agnostic of the FM type, we will focus on continuous FM for the sake of concise illustration and postpone the introduction of the discrete counterpart in Section 3.5. Recall that the aim of CFM is to train a parameterized velocity field, $v_t^\theta(\mathcal{G}_t)$ so that the sampling follows $G_{t+dt} = G_t + v_t^\theta(G_t) \cdot dt, G_0 \sim p_0(\mathcal{G})$ would generates the path along $p_t(\mathcal{G})$ and terminates at $p_1(\mathcal{G})$. We can parameterize $v_t^\theta(G_t)$ as,

$$v_t^\theta(G_t) = \mathbb{E}_{G_0 \sim p_0(\mathcal{G}), G_1 \sim p_{1|t}^\theta(\cdot,\cdot|G_t)} \left[v_t(G_t \mid G_0, G_1) \right] \tag{5}$$

As such, training the velocity function is replaced by training a denoiser $p_{1|t}^\theta(\cdot, \cdot \mid G_t)$ to predict the clean datapoint, which is commonly done by maximizing the log-likelihood [48, 10],

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{G_1 \sim p_1(\cdot), G_0 \sim p_0(\cdot), t \sim \mathcal{U}_{[0,1]}(\cdot), G_t \sim p_{t|0,1}(\cdot|G_1, G_0)} \left[\log p_{1|t}^\theta(G_1 \mid G_t) \right] \tag{6}$$

where G_1 is sampled from data distribution and G_0 from reference distribution. t is sampled from a uniform distribution $\mathcal{U}_{[0,1]}$ on $[0, 1]$. G_t can be sampled from $p_{t|0,1}$ 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. However, a closed form of $p_t(\cdot \mid G_0, G_1)$ is required to construct both the probability path and the velocity field $v_t(G_t \mid G_0, G_1)$.

3.2 Markov Random Fields for Graphs

Graph Markov Random Fields Markov random fields (MRF) is a graphical model that is commonly used in modeling interconnected systems in statistical physics, such as molecule dynamics and multi-body dynamics[3]. 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, this is described by the joint probability density distribution (PDF) of node features and graph structure can be written as

[3] see Appendix A.1 for a detailed discussion on graph generation, molecule dynamics, and MRF

$p(\mathcal{G} \mid \boldsymbol{G}) = p(\mathcal{X}, \mathcal{E} \mid \boldsymbol{X}, \boldsymbol{W}) = p(\mathcal{X} \mid \boldsymbol{X}, \boldsymbol{W}) \cdot p(\mathcal{E} \mid \boldsymbol{W})$. Instead of assuming marginal independence, the node features and graph structure are conditionally independent given latent variables $\boldsymbol{X}$ and $\boldsymbol{W}$ (As described by the graphical model in Fig. 1). For node features $\mathcal{X}$, we follow the MRF assumption in Zhu et al. [65] which considers the cliques up to rank 2, i.e., the PDF is decomposed into the node-wise $\varphi_1(v)$ and pair-wise potential $\varphi_2(u, v)$[4]:

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

with $\|\cdot\|$ the L_2 norm. It is shown in Appendix A.2 that Eq. (7) is a colored Gaussian distribution as in Eq. (8). The edges are assumed to be emitted from a Dirac function, i.e. $\mathcal{E} \sim \delta(\boldsymbol{W})$, yielding our definition for Graph Markov Random Fields (GMRF). The derivation can be found in Appendix A.2.

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

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

The $\otimes$ is the Kronecker product, $\operatorname{vec}(\cdot)$ is the vectorization operator and $\boldsymbol{I}$ is the identity matrix. The linear transformation matrix $\boldsymbol{V}$ assists in modulating the graph feature emission properties, such as transforming to approximate one-hot encodings. Equipped with a proper prior on $\boldsymbol{X}$ and $\boldsymbol{W}$, the GMRF provides an effective way to organize the graph systems.

Remark. We wish to point out that GMRF is not a universal model for all the graph models and it has constraints. The usage scope of GMRF is discussed in Appendix A.3. Fortunately, the model is capable of capturing the dynamics on most of the graph generation tasks, such as planar, SBM, and molecule graph generation.

3.3 The Optimal Transport Distance between Graph Distributions

The GMRF, simplifying all the way to Eq. (7), provides an elegant and simple framework to consider graphs as statistical objects. We further show that the paired metric spaces are equipped with a well-defined optimal transport distance, which can be used to build CFM models in Section 3.4. For such purpose, we decompose the Wasserstein distance between $\eta_{\mathcal{G}_0}$ and $\eta_{\mathcal{G}_1}$ as the sum of node feature and graph structure distance.

$$\text{(Graph Wasserstein Distance)} \quad \mathcal{W}_c\left(\eta_{\mathcal{G}_0}, \eta_{\mathcal{G}_1}\right) = \mathcal{W}_c(\eta_{\mathcal{X}_0}, \eta_{\mathcal{X}_1}) + \mathcal{W}_c(\eta_{\mathcal{E}_0}, \eta_{\mathcal{E}_1}). \tag{9}$$

where $\eta_{\mathcal{X}_j} \sim p(\mathcal{X}_j)$ is measures on node and $\eta_{\mathcal{E}_j} \sim p(\mathcal{E}_j)$ measures on edges with $j \in \{0, 1\}$.

We emphasize that $\mathcal{X}_j$ is a variable controlled by the graph structure, so the BW distance implicitly models interaction between nodes and edges. Given the the question of finding the OT distance boils down to measure the distance between two colored Gaussian distribution, we borrow the results from previous research in probability theory [14, 44, 55] that solves the Wasserstein distance between two Gaussian distributions with an analytical form introduced in Lemma 3. Combining Lemma 3 and Eq. (8) and substituting into Eq. (9), the closed form of the BW distance for graph objects are derived as following (Note that this result is an extension of [20] under the assumption of MRF),

Corollary 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 signed weighted graph Laplacian matrices $\boldsymbol{L}_0$ and $\boldsymbol{L}_1$ satisfying 1) is Positive Semi-Definite (PSD) and 2) has only one non-zero eigenvalue. The Bures-Wasserstein distance between these two random graph distributions is given by*

$$\mathcal{W}_{BW}\left(\eta_{\mathcal{G}_0}, \eta_{\mathcal{G}_1}\right) = \|\boldsymbol{X}_0 - \boldsymbol{X}_1\|_F^2 + \beta \operatorname{trace}\left(\boldsymbol{L}_0^\dagger + \boldsymbol{L}_1^\dagger - 2\left(\boldsymbol{L}_0^{\dagger 1/2} \boldsymbol{L}_1^\dagger \boldsymbol{L}_0^{\dagger 1/2}\right)^{1/2}\right), \tag{10}$$

as $\nu \to 0$ and $\beta = \|\boldsymbol{V}\|_F^2 + 1$. The proof can be found in Appendix B.2.

[4]The assumption behind, is that the observed $\mathcal{X} = \boldsymbol{V}^\dagger(\boldsymbol{\mu} + \epsilon)$ so that $\boldsymbol{V}\mathcal{X} - \boldsymbol{\mu} = \epsilon \sim \mathcal{N}(0, (\nu \boldsymbol{I} + \boldsymbol{L})^{-1})$.

3.4 Bures-Wasserstein Flow Matching for Graph Generation

With the proper definition of optimal transport distance, we are now capable of introducing the two important components for FM models - the interpolation and the velocity. This provides a closed form for the induced probability path $p(G_t \mid G_0, G_1)$ that is easy to access without any simulation.

The interpolation. We first derive the OT displacement interpolant for two graphs, which is obtained through the following displacement minimization problem,

$$\mathcal{G}_t = \arg\min_{\tilde{\mathcal{G}}} (1-t)\mathcal{W}(\eta_{\mathcal{G}_0}, \eta_{\tilde{\mathcal{G}}}) + t\mathcal{W}(\eta_{\mathcal{G}_1}, \eta_{\tilde{\mathcal{G}}}). \tag{11}$$

With the Bures-Wasserstein distance defined in Corollary 1, we prove the minimizer of the above problem has the form in Lemma 1. The proof can be found in Appendix B.3.

Lemma 1 (Bures-Wasserstein interpolation)**.** *The graph minimizer of Eq. (11) have its node features following a colored Gaussian distribution, $\eta_{\mathcal{X}_t} \sim \mathcal{N}(\boldsymbol{X}_t, \Lambda_t^\dagger)$ with $\Lambda_t = (\nu \boldsymbol{I} + \boldsymbol{L}_t) \otimes \boldsymbol{V}^\top \boldsymbol{V}$ and edges following $\mathcal{E}_t \sim \delta(\boldsymbol{W}_t)$, superficially,*

$$\boldsymbol{L}_t^\dagger = \boldsymbol{L}_0^{1/2} \left((1-t)\boldsymbol{L}_0^\dagger + t \left(\boldsymbol{L}_0^{\dagger/2} \boldsymbol{L}_1^\dagger \boldsymbol{L}_0^{\dagger/2} \right)^{1/2} \right)^2 \boldsymbol{L}_0^{1/2}, \quad \boldsymbol{X}_t = (1-t)\boldsymbol{X}_0 + t\boldsymbol{X}_1 \tag{12}$$

The velocity. We consider the reparameterization as in Eq. (5) and derive the conditional velocity $v_t(G_t \mid G_1, G_0)$ as in Lemma 2.

Lemma 2 (Bures-Wasserstein velocity)**.** *For the graph $\mathcal{G}_t$ following BW interpolation in Lemma 1, the conditional velocity at time t is given as,*

$$\begin{aligned} &v_t(E_t \mid G_0, G_1) = \dot{\boldsymbol{W}}_t = diag(\dot{\boldsymbol{L}}_t) - \dot{\boldsymbol{L}}_t, \quad v_t(X_t \mid G_0, G_1) = \boldsymbol{X}_1 - \boldsymbol{X}_0 \\ &\text{with } \boldsymbol{T} = \boldsymbol{L}_t^{1/2} (\boldsymbol{L}_t^{\dagger/2} \boldsymbol{L}_1^\dagger \boldsymbol{L}_t^{\dagger/2})^{1/2} \boldsymbol{L}_t^{1/2}, \quad \dot{\boldsymbol{L}}_t = 2\boldsymbol{L}_t - \boldsymbol{T}\boldsymbol{L}_t + \boldsymbol{L}_t\boldsymbol{T} \end{aligned} \tag{13}$$

where $\boldsymbol{W}_t = \boldsymbol{D}_t - \boldsymbol{L}_t$ and $\boldsymbol{L}_t$ defined in Eq. (12). Derivation can be found in Appendix C.3.

With Lemma 1 and Lemma 2, 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 1: Even though the GMRF in Definition 2 does rely on an implicit linear emission matrices $\boldsymbol{V}$, the BW interpolation Theorem 1 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.

Remark 2: 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 detailed discussion about the design space of BW interpolation and velocity, and compare

3.5 Discrete Bures Wasserstein Flow Matching for Graph Generation

Up to now we are working on the scenario when $p(\mathcal{X} \mid \boldsymbol{X}_i, \boldsymbol{W}_i)$ is a Gaussian and $p(\mathcal{E} \mid \boldsymbol{W}_i)$ is a Dirac distribution. However, previous studies have observed a significant improvement of the discrete counterpart of the continuous graph generation models [57, 63, 48]. To benefit our model from such a nature, we derive the discrete Bures-Wasserstein Flow Matching for graph generation.

As such, we design the probability path as,

$$p_t(\mathcal{X}) = \text{Cat}(\boldsymbol{X}_t), p_t(\mathcal{E}) = \text{Bernoulli}(\boldsymbol{W}_t) \text{ s.t. } p(\mathcal{G}_0) = \delta(G_0), p(\mathcal{G}_1) = \delta(G_1) \tag{14}$$

where $\boldsymbol{W}_t = \boldsymbol{D}_t - \boldsymbol{L}_t$ with $\boldsymbol{X}_t$ and $\boldsymbol{L}_t$ defined the same in Eq. (12). 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 $p_0(\mathcal{G}) = \delta_{G_0}(\mathcal{G}), p_1(\mathcal{G}) = \delta_{G_1}(\mathcal{G})$ holds. Even though now we are not sampling from Gaussian distributions anymore, it is possible to approximate the Wasserstein distance between two multivariate Bernoulli distributions with the Gaussian counterpart so the conclusions, such as optimal transport displacements, still hold. We left the discussion in Appendix D.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. (12);
  `/* Denoising - x-prediction */`
  $p^\theta_{1|t}(\cdot \mid G_t) \leftarrow f_\theta(G_t, t)$;
  Loss calculation via Eq. (6);
  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 $\hat{G}_1 \leftarrow p^\theta_{1|t}(\cdot \mid G_t)$;
  `/* Velocity calculation */`
  Calculate $v_\theta(\hat{G}_t \mid \hat{G}_0, \hat{G}_1)$ via Eq. (13);
  `/* Numerical Sampling */`
  Sample $\hat{G}_{t+dt} \sim \hat{G}_t + v_\theta(\hat{G}_t)dt$

The distribution of $\mathcal{X}_t$ 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 $v_t(X_t \mid G_0, G_1) = \delta(\cdot, \boldsymbol{X}_1) - \delta(\cdot, \boldsymbol{X}_0)$. However, the velocity of $\mathcal{E}_t$ cannot be written as a mixture of two boundary conditions. Instead, we derive in Appendix C.3 that the discrete velocity as,

$$v_t(E_t \mid G_1, G_0) = (1 - 2E_t)\,\frac{\dot{\boldsymbol{W}}_t}{\boldsymbol{W}_t \circ (1 - \boldsymbol{W}_t)}, \tag{15}$$

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. (12) and (13) respectively. With the interpolation and velocity defined, the discrete flow matching is built in Algorithms 3 and 4.

4 Experiments

We evaluate the BWFlow algorithm 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 on both tasks in Section 4.2. Next, we conduct ablation studies to analyze the impact of interpolation methods and the corresponding velocity construction on graph generation performance in Section 4.3 and demonstrate the effectiveness and benefit of flow along Bures-Wasserstein interpolation.

4.1 Experiment Settings

Dataset. For planar graph generation, we evaluate the quality of generated graphs on three benchmark datasets following previous works [38, 57, 6], specifically, **planar** graphs, **tree** graphs, and stochastic blocking models (**SBM**). Two datasets, namely MOSES [46] and GUACAMOL [8], are benchmarked to test the model performance on 2D molecule generation. For 3D molecule generation with coordinate data, we test the model on QM9 [49] 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 L.1). For molecule generation, since our model ignores the edge types but only consider the graph structure when generating the graphs, we develop a new relaxed metric when measuring the stability and validity of atoms and molecules. Specifically, the atom-wise stability is relaxed as:

$$\text{Stability of Atom } i: \quad s_i = \mathbb{I}[\exists (b_{ij})_{j \in \mathcal{N}_i} \in \prod_{j \in \mathcal{N}_i} B_{ij} : \sum_{j \in \mathcal{N}_i} b_{ij} = E_i] \tag{16}$$

which 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 (E_i). Such a relaxed stability of atoms (**Atom.Stab.**) inherently defines molecule stability (**Mol.Stab.**) and the **V.U.N.** 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, angels, etc.). Details in Appendix L.1.

Setup. To isolate the impact from model architecture, we follow [48] to fix the backbone model as the same graph transformers in their paper. Furthermore, it is shown in [48] that sample distortion, training distortion and target guidance have a significant impact on the performance of graph generation tasks. In our paper, in addition to the best model performance, we also show the results when these technologies are disabled, i.e., identity time distortion and unconditional generation without any graph-level labeling.

4.2 Main Results for Graph Generation

Plain Graph Generation. We first validate the ability of BWFlow to generate plain graphs without node attributes. In Table 1, we report V.U.N. and A.Ratio. We only keep the main diffusion/flow model for comparison, while other models are included in the full version at Table 6. As the model performance is near-saturated in these datasets, we not only report the best performance achieved by the models but also gives the results when all the models are without training distortion and sampling distortion as in Qin et al. [48]. One exception is tree graph datasets, where our model performance is not satisfying. This is due to the fact that tree graphs lie in a hyperbolic space which do not strictly follow our MRF assumptions.

Table 1: 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.

		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 [57]	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 [6]	Diffusion	95.0	2.1	**100.0**	4.0	75.0	10.5
GruM [27]	Diffusion	90.0	1.8	—	—	85.0	**1.1**
CatFlow [16]	Flow	80.0	—	—	—	85.0	—
DisCo [63]	Diffusion	83.6±2.1	—	—	—	66.2±1.4	—
Cometh [50]	Diffusion	**99.5**±0.9	—	—	—	75.0±3.7	—
DeFoG [48]	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	**1.37**±0.4	75.5±2.4	**3.6**±1.2	83.5±6.0	3.8±0.9
DeFoG (S.ed)	Flow	77.5±1.0	3.5±0.4	75.3±2.6	1.32±0.4	**90.0**±5.1	4.9±1.3
BWFlow (S.ed)	Flow	**84.8**±1.0	**2.7**±0.4	75.5±2.4	**3.6**±1.2	**90.5**±4.0	3.8±0.9

2D Molecule Graph Generation The model performance compared to SOTA diffusion/flow models is illustrated in Table 2 In GUACAMOL, almost all the modern generative models, including our BWflow, can achieve V.U.N close to 100%. 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.

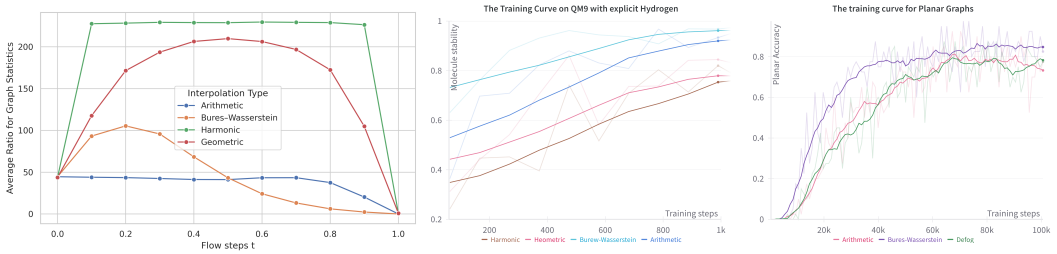
Table 2: Large molecule generation results. Only iterative denoising-based methods are reported.

	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 [57]	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 [63]	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 [50]	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 [48]	**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			91.7	91.7	91.2	92.3	57.9		

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 fair comparison. Interestingly, the empirical results show that even without edge type, the de novo graph generation model already can capture the molecule data distribution. And our BWFlow significantly

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		35.7		/	/	/		
	BWFlow	1.20	87.75	**46.80**	**95.08**	73.53	**0.1**	**6.5**	**3.96**

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



(a) The evolution of graph statistics ratio. (b) Training Curves for QM9. (c) Training Curves for Planar.

Figure 2: Training curves on QM9 and planar datasets with explicit hydrogen.

outperforms the SOTA models, including MiDi [58] and FlowMol [15]. We believe a promising future direction is to incorporate the processing of multiple bond types into our framework, which would further raise the performance by a margin.

4.3 Ablation Studies.

Superiority of BWFlow in constructing paths. We compute the A.Ratio on SBM between generated graph interpolants and test data for $t \in [0, 1]$, as shown in Fig. 2a. Under arithmetic interpolation, the A.Ratio hovers around a high value until $t \approx 0.8$. By contrast, BW interpolation initially exposes the model to more out-of-distribution samples with increased A.Ratio. After this early exploration, the A.Ratio rapidly converges, yielding more accurate velocity estimates toward the true data distribution. This behavior—early exposure to diverse samples followed by steady denoising—enhances both the model robustness and velocity estimation. In comparison, harmonic and geometric interpolations step outside the valid graph domain, making the learning problem ill-posed.

Interpolation metrics

An interesting thing that we observe, is that our methods can improve the convergence and performance to some extend except that when the dealing with tree graph. In tree graphs, simply using arithmetic mean can perform very well in conditional flow matching experiments. Why is that?

5 Discussion and Future Work

Limitations: *Extension to multiple relation types.* As our framework is built upon the interpolation parameterized by the Laplacian Matrices, it is not generalizable to the probability path with multiple edge types. *Increased Computational Complexity.* While constructing the probability path and the velocity, our BW interpolation suffers from an $O(N^3)$ extra cost due to its request to compute the pseudo-inverse of the Laplacian matrix. Compared to linear interpolation, empirically this brings 2x training time and inference time.

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] Albert-László Barabási, Réka Albert, and Hawoong Jeong. Mean-field theory for scale-free random networks. *Physica A: Statistical Mechanics and its Applications*, 272(1-2):173–187, 1999.

[5] Olivier A Bauchau. Computational schemes for flexible, nonlinear multi-body systems. *Multibody System Dynamics*, 2(2):169–225, 1998.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[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] 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.

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

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

[17] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 12 2007. ISSN 1465-4644. doi: 10.1093/biostatistics/kxm045. URL https://doi.org/10.1093/biostatistics/kxm045.

[18] 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.

[19] Zoubin Ghahramani. An introduction to hidden markov models and bayesian networks. *International journal of pattern recognition and artificial intelligence*, 15(01):9–42, 2001.

[20] 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.

[21] 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.

[22] Yilin He, Xinyang Liu, Bo Chen, and Mingyuan Zhou. Advancing graph generation through beta diffusion. *CoRR*, abs/2406.09357, 2024.

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

[24] 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.

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

[26] 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=JHKOQBKdYY.

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

[28] Vassilis Kalofolias. How to learn a graph from smooth signals. In Arthur Gretton and Christian C. Robert, editors, *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 920–929, Cadiz, Spain, 09–11 May 2016. PMLR.

[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, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard S. Zemel. Neural relational inference for interacting systems. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 2693–2702. PMLR, 2018.

[31] I.N. Levine. *Quantum Chemistry*. Pearson advanced chemistry series. Pearson, 2014. ISBN 9780321890603. URL https://books.google.co.uk/books?id=ht6jMQEACAAJ.

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

[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] Jie Liu, Chunming Zhang, Elizabeth Burnside, and David Page. Learning Heterogeneous Hidden Markov Random Fields. In Samuel Kaski and Jukka Corander, editors, *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, volume 33 of *Proceedings of Machine Learning Research*, pages 576–584, Reykjavik, Iceland, 22–25 Apr 2014. PMLR. URL https://proceedings.mlr.press/v33/liu14.html.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] Leonie Neuhäuser, Andrew Mellor, and Renaud Lambiotte. Multibody interactions and nonlinear consensus dynamics on networked systems. *Physical review. E*, 101 3-1:032310, 2019. URL https://api.semanticscholar.org/CorpusID:204800504.

[42] 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.

[43] Frank Noé and Hao Wu. Boltzmann generators - sampling equilibrium states of many-body systems with deep learning. *CoRR*, abs/1812.01729, 2018.

[44] 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.

[45] Sungwoo Park, Dongjun Kim, and Ahmed Alaa. Mean-field chaos diffusion models. In *ICML*. OpenReview.net, 2024.

[46] 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.

[47] 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.

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

[49] 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.

[50] 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.

[51] 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.

[52] 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.

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

[54] A. Szabo and N.S. Ostlund. *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory*. Dover Books on Chemistry. Dover Publications, 1996. ISBN 9780486691862. URL https://books.google.co.uk/books?id=6mV9gYzEkgIC.

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

[56] 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.

[57] 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.

[58] 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.

[59] 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.

[60] 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.

[61] Rongzhe Wei, Haoteng Yin, Junteng Jia, Austin R. Benson, and Pan Li. Understanding non-linearity in graph neural networks from the bayesian-inference perspective. In *NeurIPS*, 2022.

[62] 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.

[63] 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.

[64] 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.

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

[66] 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.