Computer Science Review 59 (2026) 100854

Contents lists available at ScienceDirect

Computer Science Review

journal homepage: www.elsevier.com/locate/cosrev

Review article

Graph diffusion models: A comprehensive survey of methods and applications

Yuntao Shou [a], Wei Ai [a], Tao Meng [a,*], Keqin Li [b]

[a] College of Computer and Mathematics, Central South University of Forestry and Technology, 410004, Hunan, Changsha, China
[b] Department of Computer Science, State University of New York, New Paltz, NY, 12561, USA

ARTICLE INFO

Keywords:
Graph diffusion models
Generative models
Graph neural network
Graph generation

ABSTRACT

Diffusion models have rapidly emerged as a new paradigm in generative modeling. Therefore, we aim to provide a comprehensive review of graph diffusion models. We introduce various forms of diffusion models (i.e., DDPMs, SDEs, and SGMs), their working mechanisms, and how they can be extended to graph data. Specifically, graph diffusion models follow the modeling process of diffusion models, implement the diffusion process in graph data, and gradually denoise and generate new graph structures through reverse steps. The application of graph diffusion models is mainly focused on the application scenarios of generating molecules and proteins, but graph diffusion models also show potential in recommendation systems and other fields. We explore the performance and advantages of graph diffusion models in these specific applications, such as using them to discover new drugs and predict protein structures. Furthermore, we also discuss the problem of evaluating graph diffusion models and their existing challenges. Due to the complexity and diversity of graph data, the authenticity of generated samples is an important and challenging task. We analyze their limitations and propose potential improvement directions to better measure the effectiveness of graph diffusion models. The summary of existing methods mentioned is in our Github: https://github.com/yuntaoshou/Graph-Diffusion-Models.

Contents

* Corresponding author.
E-mail address: mengtao@hnu.edu.cn (T. Meng).

https://doi.org/10.1016/j.cosrev.2025.100854
Received 24 June 2025; Received in revised form 15 September 2025; Accepted 4 November 2025
Available online 10 November 2025

1. Introduction

In the past two years, diffusion methods [1–3] in image generation have received widespread attention in graph generation, resulting in a significant increase in research activities on graph diffusion models and the emergence of more and more new methods and technologies [4–18]. As shown in Fig. 1(a), Diffusion models in the image generation generate data by gradually applying noise and denoising processes [19]. The basic idea is to gradually transform data into pure noise through multiple steps of noise, and then gradually denoise through the reverse process to restore meaningful data. As shown in Fig. 1(b), Diffusion models [20–23] for graph generation follow a similar paradigm. The process involves adding noise to a graph's structure and features over several steps (forward process) and learning to denoise noisy graphs back to their original form (reverse process) [24]. Although graph diffusion models have developed rapidly, they have also caused researchers, especially new entrants, to face the problem of information overload. New researchers may find it difficult to screen out the most influential work and face challenges in understanding and applying new methods. To help researchers understand and distinguish different methods, there are several survey papers for graph diffusion generation [25,26]. However, most of them focus on the description of the current research status and lack mathematical theoretical analysis of subsequent research progress.

Therefore, based on the above analysis, our review is the first to comprehensively review the current status of graph diffusion model research, covering the current main research directions and the latest developments. As shown in Table 1, our review divides the graph generation methods into four categories, i.e., traditional graph generation methods, which include four subcategories, i.e., autoregressive models (AR) [27], normalizing flows [28], VAE [29], and GAN [30], and graph diffusion methods, which include three subcategories, i.e., DDPMs [20], SGMs [19,31], and SDEs [32,33]. We discuss the theoretical basis, practical applications, advantages, and disadvantages of different methods so that readers can better understand the applicable scenarios and limitations of each method. We will focus on highlighting the key progress and breakthroughs in the field of graph diffusion models including theoretical innovations, improvements in empirical performance, and important results in practical applications. We hope that this review can provide a useful point for researchers in the field of graph generation, allowing them to quickly understand the basic concepts, latest progress, and research trends of graph diffusion models. Meanwhile, we also hope to provide experienced researchers with a broader perspective to help them identify potential challenges.

The contributions of this paper are summarized as follows:

- **Comprehensive Review:** We provide the most comprehensive review of traditional models (i.e., AR, normalizing flows, VAEs, and GANs) and graph diffusion models (i.e., DDPMs, SGMs, and SDEs) for graph generation.
  Insightful Analysis. For each modeling approach, we provide a representative model and give a mathematical theoretical derivation, which can guide readers to choose an appropriate baseline model for their research.
- **Abundant Resources:** We have collected relevant resources on graph generation, including SOTA models and publicly available datasets on Github.[1] This paper can serve as a practical guide for learning and developing different graph generation algorithms.
- **Future Directions:** We analyze the limitations of existing graph diffusion methods and propose possible future research directions from multiple aspects, including training objectives, scalability from 2D to 3D graph generation, and data distribution.

The paper is organized as follows: Section 2 summarizes traditional graph generation methods. Section 3 illustrates the background, definitions for graph diffusion generation, and divides graph diffusion generation methods into three categories, and gives the mathematical theory. Section 4 summarizes some of the publicly available and popular datasets for graph generation tasks, and Section 5 gives the performance of different algorithms. Section 6 discusses the great value and broad prospects of graph diffusion models in practical applications. Section 7 illustrates the future research directions and challenges of graph diffusion models. Finally, we conclude the work.

2. Traditional graph generation methods

In this section, we briefly review traditional graph generation methods. We roughly divide the existing traditional graph generation methods into four categories, i.e., autoregressive models (AR) [27], normalizing flows [28], variational autoencoders (VAE) [29], and generative adversarial networks (GAN) [30].

2.1. Autoregressive models

The basic idea of AR [27] is to decompose the graph generation process into a series of decisions, each step is based on the previous decision. First, AR needs to determine the order of generating nodes, which can be a random order, the order of node degrees, or other heuristic methods. Then generate nodes one by one in the determined order. The generation of each node can be based on the information of the previously generated nodes. For each generated node, decide in turn whether there is an edge between it and all previous nodes. This

[1] https://github.com/yuntaoshou/Graph-Diffusion-Models-A-Comprehensive-Survey-of-Methods-and-Applications

Table 1
We summarize variants of traditional graph generation methods and variants of diffusion models.

Categories		Year	Methods
Traditional Methods	Auto-regressive	2018	GraphRNN [34]
		2019	MolecularRNN [35]
		2020	ES-GraphRNN [36], GraphGen [37]
		2021	GraphGen-Redux [38]
	VAE	2018	DCRM [39], JT-VAE [40]
		2019	ConGen [41], D-VAE [42]
		2020	IMGDL [43], PCVAE [44], NECD [45], CBO-VAE [46], NeVAE [47], DGVAR [48]
		2021	DLM [49], COVAE [50], DDGSN [51]
		2022	IMGMC [52], DSG [53]
	Normalizing Flows	2019	GraphNVP [54]
		2020	MoFlow [55]
		2021	GraphDF [56]
	GAN	2018	LMGT [57]
		2019	CLGAN [58]
		2020	Mol-CycleGAN [59], TSGG-GAN [60]
		2021	ALMGIG [61]
Diffusion Models	DDPMs	2022	EDM [4], GeoDiff [5], DiffBP [62], DiffSBDD [63], DiffAb [64], Anand [65], PROTSEED [66]
		2023	MDM [67], Digress [68], TargetDiff [69], DIFFDOCK [70], SILVR [71], SMCDiff [72], HierDiff [73]
		2023	RINGER [74], GeoLDM [75], Diffmol [76], DecompDiff [77], HouseDiffusion [78], DiffSTG [79]
		2023	EDGE [24], EDGE++ [80], MCRDiff [81], SaGess [82], DIFUSCO [83]
		2024	MDM, MiDi [84], GCDM [85], PMDM [86], TSDiff [87]
		2024	DiffLinker [6], D3FG [88], GradeIF [89], GFMDiff [90], MUDiff [91], Pard [92]
		2024	AbDiffuser [93], SPDiff [94], LatentDiff [95], DiffCSP [96], DiffCSP++ [97], HypDiff [98]
		2024	GemsDiff [99], CrysDiff [100], CHP-MOFassemble [101], NAP [102], DDM [103], ILE [104]
		2025	GBD [105], DiffGraph [106]
	SGMs	2020	EDP-GNN [107]
		2021	ConfGF [108], ColfNet [109], DGSM [110], ProteinSGM [111]
		2022	DiffPB [112], SLD [113]
		2023	DiffusionCG [114], DruM [115]
		2024	VoxMol [116]
	SDEs	2022	EEDSDE [117], GraphGDP [118], NVDiff [119]
		2023	CDGS [120], DiffMD [121], EigenFold [122], JODO [123], MuDM [124]
		2023	GSDM [125], DiffusionNAG [126], Diff-POI [127]
		2024	NeuralPLexer [128], SD [129], DiffBindFR [130], HGDM [131], Graphhusion [9], ProGDM [132]

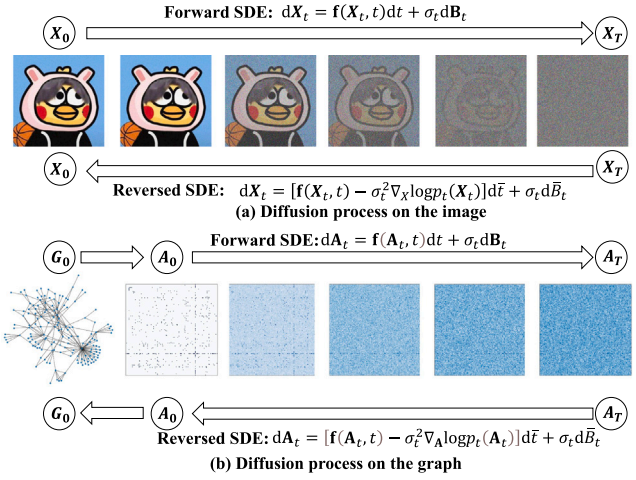

Fig. 1. The workflow of the diffusion model on images and graphs. The arrows on the right indicate the direction of the diffusion process, showing how noise is gradually injected into the real data to simulate the transition from order to disorder. In this stage, the real data is gradually added with noise through multiple iterations and finally becomes a pure noise distribution. Guided by the arrows on the left, the sampling stage of generating samples is shown. This stage reverses the previous diffusion process, starting from pure noise, gradually removing noise through multiple iterations and regenerating samples. This process can be seen as a transition from disorder to order, gradually recovering the generated samples close to the real data through multiple iterations and calculations.

decision can be made through conditional probability which is calculated based on the part of the graph that has been generated. In existing work, GraphRNN [34] is an autoregressive model based on RNN, which models the graph generation process as a sequence generation process of nodes and edges, and uses RNN to capture the structural information. ESGraphRNN [36] models the graph generation process as an edge-by-edge prediction process. ESGraphRNN is trained by maximizing the likelihood estimate, and the model learns to maximize the probability of generating the graph. The teacher forcing strategy is used in the training process, that is, the real previous edge information is used at each step of generation. Autoregressive methods are usually suitable for generating graph structures with fixed order and dependencies. For graphs without obvious order or irregular structure, the effect of autoregressive methods may not be ideal [133].

2.2. Normalizing flows

Normalizing Flows [28] is a technique that gradually transforms a simple distribution (usually a standard normal distribution) into a complex distribution. It maps simple distribution samples into complex distributions through a series of reversible transformations. In graph generation tasks, Normalizing Flows maps simple prior distributions (such as Gaussian distributions) into graph structure distributions through a series of reversible transformations, thereby achieving graph generation. For example, GraphNVP [54] is a model that applies Normalizing Flows to molecular graph generation. Through a series of reversible transformations, simple distributions are mapped into complex distributions of molecular graphs, thereby achieving efficient molecular graph generation. MoFlow [55] proposes a new reversible flow model for graph generation and optimization. Molecular generation is achieved by mapping the attributes of molecular graphs into latent

space and modeling them through flow models. The training of Normalizing Flows involves the optimization of log-likelihood estimation, and the training process may be unstable, especially when dealing with high-dimensional data, which may cause problems such as numerical instability and gradient vanishing [56,134].

2.3. Variational autoencoders

In graph generation tasks, VAE [29,135] encodes the structural information of the graph into the latent space and then decodes it to generate a new graph structure. For example, GraphVAE [136] uses the variational autoencoder framework to combine the node and edge attribute information of the graph to achieve small-scale graph generation. MolVAE [137] is a variational autoencoder model which encodes the structural information of the molecular graph into the latent space and generates a new molecular graph through the decoder, the molecular structure is effectively generated. To achieve sampling from the latent space and gradient transfer, MolVAE uses the reparameterization technique. By decomposing the sampling process of the latent variables into deterministic variables and random noise, effective gradient transfer can be achieved. The reconstruction quality of VAE is usually lower than that of other generative models, e.g., GAN [30]. This is because VAE needs to balance the reconstruction loss and KL divergence regularization during training, resulting in the generated graph quality may not be high [42,47].

2.4. Generative adversarial networks

GAN [30] is consisting of a generator and a discriminator. The generator samples from a noise vector (e.g., Gaussian distribution or uniform distribution) to generate the node and edge structures of the graph. The discriminator takes the real graph and the generated graph as input and outputs a probability, indicating the probability that the input graph is the real graph. For example, NetGAN [138] uses random walks to generate sequences, which are then used to generate graphs. The generator generates the local structure of the graph through random walks, and the discriminator distinguishes between the real and generated random walk sequences. MolGAN [139] is a GAN model for molecular graph generation. The generator generates the node and edge properties of the molecular graph, and the discriminator determines whether the generated molecular graph is similar to the real molecular graph. In addition, MolGAN also uses a reward network to optimize the generator for specific goals (e.g., drug activity). The training process of GAN is prone to instability. Imbalanced training of the generator and the discriminator may lead to poor generation quality or mode collapse, i.e., the graphs generated by the generator lack diversity [41].

3. Background on diffusion models

Diffusion models are a type of generative model that destroys the data distribution by gradually adding noise, and then restores the data distribution through a reverse denoising process. Diffusion models have received widespread attention in areas such as image generation and text generation. There are three main subtypes of diffusion models: DDPM, SGMs, and SDEs.

3.1. Denoising diffusion probabilistic models (DDPMs)

DDPM [20] is a type of generative model that corrupts data by gradually adding noise, and then reconstructs the data by gradually denoising it through a learning reverse process. In DDPM, the forward process is a fixed Markov chain that starts with the original data and gradually adds Gaussian noise until it becomes pure noise.

Forward process. The forward process aims to transform the input data into a standard Gaussian distribution by gradually adding Gaussian noise [67,84]. Specifically, for a data distribution $m_0 \sim q(m_0)$, the forward process is a Markov chain defined between data m_0 and m_T, which can be formally defined as:

$$\begin{aligned} q(m_{1:T}|m_0) &:= \prod_{t=1}^{T} q(m_t|m_{t-1}), \\ q(m_t|m_{t-1}) &:= \mathcal{N}(m_t; \sqrt{1-\beta_t}m_{t-1}, \beta_t I) \end{aligned} \tag{1}$$

where β is the variance of the noise added at each time step. Assume $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=0}^{t} \alpha_s$, we can get:

$$q(m_t|m_0) := \mathcal{N}(m_t; \sqrt{\bar{\alpha}_t}m_0, (1-\bar{\alpha}_t)I) \tag{2}$$

Reverse process. The reverse process uses the trained network to predict the noise added in the forward process [62,85]. Specifically, first, an initial noise sample m_T is sampled from the distribution $p(T)$. Next, the noise added is gradually removed using the trained neural network. Through the above steps, the network is trained to restore the sample m_t to m_{t-1}.

$$p_\theta(m_{t-1}|m_t) = \mathcal{N}(m_{t-1}; \mu_\theta(m_t, t), \Sigma_\theta(m_t, t)) \tag{3}$$

where $\mu_\theta(m_t, t)$ is the mean function, which calculates the mean from m_t recovered to m_{t-1}. $\Sigma_\theta(m_t, t)$ is the variance function of the neural network output, which is used to calculate the variance from m_t to m_{t-1}. By minimizing the reconstruction error from m_T to m_0, we can optimize the neural network to generate new samples consistent with the true data distribution $q(m_0)$.

The training objective is to maximize $p_\theta(m_0)$, which can be achieved by minimizing its negative log-likelihood [63,70]. The variational lower bound can be decomposed into a series of KL divergence terms:

$$\begin{aligned} \mathcal{L}(\theta) = \mathbb{E}_q \Big[&-\log p_\theta(m_0|m_1) \\ &+ \sum_{t=2}^{T} \mathrm{KL}(q(m_{t-1}|m_t, m_0) \parallel p_\theta(m_{t-1}|m_t)) \\ &+ \mathrm{KL}(q(m_T|m_0) \parallel p(m_T)) \Big] \end{aligned} \tag{4}$$

In practice, the above objectives are often simplified. A common simplified objective is to directly optimize the reconstruction error (i.e., the prediction error of the denoising process), which can be achieved by rewriting the KL divergence term and simplifying the variance assumption [1,80]. Therefore, the final optimization objective can be formally defined as follows:

$$\mathcal{L}(\theta) = \mathbb{E}_{q(m_t|m_0)} \left[\| \epsilon - \epsilon_\theta(m_t, t) \|^2 \right] \tag{5}$$

where ϵ is the added Gaussian noise and ϵ_θ is the noise predicted by the neural network.

DDPMs [20] for graph generation follow a similar paradigm. The process involves adding noise to a graph's structure and features over several steps (forward process) and learning to denoise noisy graphs back to their original form (reverse process). This probabilistic framework allows for the generation of complex graph structures with high fidelity. Next we summarize several representative works on graph generation using DDPMs.

3.1.1. Equivariant diffusion model (EDM)

As shown in Fig. 2, EDM [4] is a generative model specifically designed to preserve symmetries or invariants in the data (e.g., rotational or translational invariance). EDM is particularly useful in the molecular graph generation, where the properties of the generated structures must remain invariant under symmetry transformations. We define the coordinate m_i with atomic features h_i as an equivariant diffusion process that adds noise to the data and preserves symmetry. Assume a set of points (m_i, h_i) with spatial location information and feature information, where each node is associated with its coordinate representation $m_i \in \mathbb{R}^3$ and attribute vector $h_i \in \mathbb{R}^d$. The forward equivariant noise process for the latent feature vectors $z_t = [z_t^m, z_t^h]$ is defined as follows:

$$q(\boldsymbol{z}_t|\boldsymbol{m}, \boldsymbol{h}) = \mathcal{N}_{mh}(\boldsymbol{z}_t|\alpha_t[\boldsymbol{m}, \boldsymbol{h}], \sigma_t^2 \boldsymbol{I}) \tag{6}$$

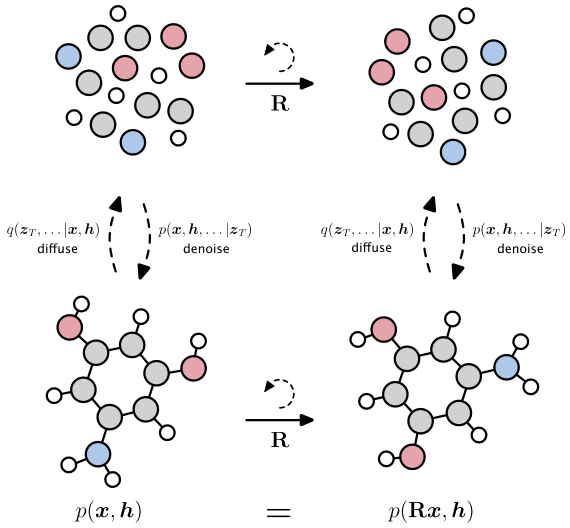

Fig. 2. An example of the overall process of EDM [4]. Through a step-by-step denoising process, EDM transforms a normally distributed random point set into a molecular structure with specific 3D coordinates and atomic types. The rotational isovariance ensures the direction independence of the generated molecules, so that the generated molecules still maintain the stability of their structure and generation probability when rotating in space.

where $\mathcal{N}_{mh}$ is the product of the noise coordinate distribution $\mathcal{N}_m$ and the noise feature distribution $\mathcal{N}_h$. $\mathcal{N}_h$ is defined as follows:

$$\mathcal{N}_m(\boldsymbol{z}_t^{(m)}|\alpha_t\boldsymbol{m},\sigma_t^2\boldsymbol{I})\cdot\mathcal{N}(\boldsymbol{z}_t^{(h)}|\alpha_t\boldsymbol{h},\sigma_t^2\boldsymbol{I}) \tag{7}$$

To denoise during the generation process, we define a denoising process based on the noise posterior distribution $q(z_s|m,h,z_t)$, where the data variables m,h are replaced by the approximation $\hat{m},\hat{h}$ of the neural network. The core idea of the denoising process is to estimate the parameters in the noise posterior distribution, thereby effectively removing the noise and restoring the original data [1,2]. Specifically, the denoising process is defined as follows:

$$p(\boldsymbol{z}_s|\boldsymbol{z}_t)=\mathcal{N}_{mh}(\boldsymbol{z}_s|\boldsymbol{\mu}_{t\to s}([\hat{\boldsymbol{m}},\hat{\boldsymbol{h}}],\boldsymbol{z}_t),\sigma_{t\to s}^2\boldsymbol{I}) \tag{8}$$

The generative denoising process relies on the intermediate variable z_t, and the output predictions of the neural network ϕ to gradually remove the noise. In existing diffusion models [62,85,86], it is common to use noise parameterization to obtain the final predictions $\hat{m}$ and $\hat{h}$. Specifically, the model does not directly output these predictions, but outputs parameter estimates related to the noise. From the parameter estimates, we can infer the actual predictions. In other words, the network ϕ does not directly predict the final form of the data, but outputs noise estimates $\hat{\epsilon}=[\hat{\epsilon}^{(m)},\hat{\epsilon}^{(h)}]$. This includes an estimate of the noise of the input data at the current time step t. The calculation process of $\hat{m},\hat{h}$ is as follows:

$$[\hat{\boldsymbol{m}},\hat{\boldsymbol{h}}]=\boldsymbol{z}_t/\alpha_t-\hat{\epsilon}_t\cdot\sigma_t/\alpha_t \tag{9}$$

The final optimization objective can be formally defined as follows:

$$\mathcal{L}_t=\mathbb{E}_{\epsilon_t\sim\mathcal{N}_{mh}(0,\boldsymbol{I})}\left[\frac{1}{2}w(t)\parallel\epsilon_t-\hat{\epsilon}_t\parallel^2\right] \tag{10}$$

where $w(t)=(1-\frac{\alpha_{t-1}}{1-\alpha_{t-1}})/\frac{\alpha_t}{1-\alpha_t}$ and $\hat{\epsilon}_t=\phi(\boldsymbol{z}_t,t)$.

3.1.2. Discrete denoising diffusion (DiGress)

DiGress [68] is mainly used to generate discrete structured data (e.g., molecular graphs, social networks, etc.) The main idea of DiGress is to use the diffusion process to decompose the complex graph generation task into multiple simple step-by-step generation steps. By gradually removing noise during the graph generation process, the generated graph structure is ensured to be valid and have the target attributes [64]. Similar to the diffusion model for images, DiGress performs diffusion on each node and edge feature separately. For node type $\mathcal{X}$ and edge type $\mathcal{E}$, the transition probabilities are defined as $[\boldsymbol{P}_{\mathcal{X}}^t]_{ij}=q(m^t=j|m^{t-1}=i)$ and $[\boldsymbol{P}_{\mathcal{E}}^t]_{ij}=q(e^t=j|e^{t-1}=i)$. At each time step t, we form $G^t=(\boldsymbol{\mathcal{X}}^t,\boldsymbol{\mathcal{E}}^t)$ by adding noise to the nodes and edges. DiGress samples from the categorical distribution defined as:

$$\begin{aligned}q(\mathcal{G}^t|\mathcal{G}^{t-1})&=(\boldsymbol{\mathcal{X}}^{t-1}\boldsymbol{P}_{\mathcal{X}}^t,\boldsymbol{\mathcal{E}}^{t-1}\boldsymbol{P}_{\mathcal{E}}^t)\\ q(\mathcal{G}^t|\mathcal{G})&=(\boldsymbol{\mathcal{X}}\bar{\boldsymbol{P}}_{\mathcal{X}}^t,\boldsymbol{\mathcal{E}}\bar{\boldsymbol{P}}_{\mathcal{E}}^t)\end{aligned} \tag{11}$$

where $\boldsymbol{P}_{\mathcal{X}}^t=\{\boldsymbol{P}_{\mathcal{X}}^1\dots\boldsymbol{P}_{\mathcal{X}}^t\}$ and $\bar{\boldsymbol{P}}_{\mathcal{E}}^t=\{\boldsymbol{P}_{\mathcal{E}}^1\dots\boldsymbol{P}_{\mathcal{E}}^t\}$. Then DiGress inputs the noisy graph $\mathcal{G}_t=(\boldsymbol{\mathcal{X}}_t,\boldsymbol{\mathcal{E}}_t)$ into the denoising neural network ϕ_θ, where $\boldsymbol{\mathcal{X}}_t$ represents the noise node feature, and $\boldsymbol{\mathcal{E}}_t$ represents the noise edge feature. The neural network ϕ_θ predicts the clean node and edge features $(p^{\mathcal{X}},p^{\mathcal{E}})$ through calculation. To make the predicted value $(p^{\mathcal{X}},p^{\mathcal{E}})$ closer to the true value $(\boldsymbol{\mathcal{X}},\boldsymbol{\mathcal{E}})$, DiGress uses the cross-entropy (CE) loss to measure the difference as follows:

$$l(\hat{p}^{\mathcal{G}},\mathcal{G})=\sum_{1\le i\le n}\mathrm{CE}(m_i,\hat{p}_i^{\mathcal{X}})+\lambda\sum_{1\le i,j\le n}\mathrm{CE}(e_{ij},\hat{p}_{ij}^{E}) \tag{12}$$

where $\lambda\in\mathbb{R}^+$ is a hyperparameters. When the denoising network ϕ_θ is trained, DiGress uses it to sample new graphs. Specifically, DiGress models the distribution of nodes and edges as follows:

$$p_\theta(\mathcal{G}^{t-1}|\mathcal{G}^t)=\prod_{1\le i\le n}p_\theta(m_i^{t-1}|\mathcal{G}^t)\prod_{1\le i,j\le n}p_\theta(e_{ij}^{t-1}|\mathcal{G}^t) \tag{13}$$

To calculate each term, DiGress marginalizes these probability distributions. Specifically, for each node and edge feature, DiGress calculates the edge probability under all possible states as follows:

$$p_\theta(m_i^{t-1}|\mathcal{G}^t)=\sum_{m\in\mathcal{X}}p_\theta(x_i^{t-1}\mid m_i=m,\mathcal{G}^t)\hat{p}_i^{\mathcal{X}}(m) \tag{14}$$

where all possible states of $\mathcal{G}_{t-1}$ are combined with the current predicted probability $\hat{p}^{\mathcal{G}}$ and summed to obtain the marginalized probability distribution, and

$$p_\theta(m_i^{t-1}\mid\mathcal{G}^t)=\begin{cases}q(m_i^{t-1}\mid m_i=m,m_i^t) & \text{if}\quad q(m_i^t|m_i=m)>0\\ 0 & \text{otherwise.}\end{cases} \tag{15}$$

Similarly, $p_\theta(e_{ij}^{t-1}|e_{ij}^t)=\sum_{e\in\mathcal{E}}p_\theta(e_{ij}^{t-1}\mid e_{ij}=e,e_{ij}^t)\hat{p}_{ij}^{\mathcal{E}}(e)$.

3.1.3. Diffusion with discrete state spaces (DDSS)

DDSS [140] focuses on the generation process of simple graphs. Through the diffusion model represented by discrete processes, it can effectively simulate the generation process of graph structures and provide high-quality generated samples in various applications [65,87]. The core of DDSS lies in the effective combination of forward and reverse processes, which enables the generation model to not only add noise, but also effectively remove noise, thereby generating high-quality samples that meet the target distribution.

In the graph generation model, DDSS first defines a one-hot encoded row vector a_{ij}^t for element (i,j) in the adjacency matrix Θ_t. This vector belongs to the set $\{0,1\}^2$. The initial adjacency matrix Θ_0 is a sample directly sampled, and Θ_T represents an Erdős–Rényi random graph [141]. The forward process implements the stepwise noise addition process from A_0 to A_T by repeatedly multiplying the type row vector $q(\boldsymbol{a}_t^{ij}|\boldsymbol{a}_t^{ij})=\mathrm{Cat}(\boldsymbol{a}_t^{ij}|p=\boldsymbol{a}_{t-1}^{ij}\boldsymbol{Q}_t)$ of each adjacency matrix element with the doubly random matrix Q_t. Cat represents a categorical distribution, and its parameter p is determined by the product of the one-hot encoded row vector a_{ij}^{t-1} of the previous time step and the matrix Q_t. In the forward process, the operation of each edge or non-edge $i\neq j$ is independent, which means that we can process these elements in parallel, thereby improving computational efficiency. The matrix Q_t is a doubly random matrix with a dimension of $\mathbb{R}^{2\times2}$, which is used to model the transition probability of adjacency matrix elements between different time steps. Specifically, Q_t is defined as:

$$\boldsymbol{Q}_t=\begin{bmatrix}1-\delta_t & \delta_t\\ \delta_t & 1-\delta_t\end{bmatrix} \tag{16}$$

where δ_t represents the invariant probability of the edge state. An important advantage of Eq. (16) is that it can be sampled directly of the diffusion process without relying on the calculation of any previous time step. Formally, δ can be replaced by $\delta_t = \frac{1}{2} - \frac{1}{2}\prod_{i<t}(1-2\delta_i)$. DDSS limits the probability of δ_t to vary from 0 (undisturbed samples) to 0.5 (noise), ensuring that the diffusion process retains more original information in the initial stage and tends to complete randomness in the final stage. In particular, the noise is sampled in i.i.d. manner on all edges.

Since the forward process is a Markov chain [20], the current state $\boldsymbol{\Theta}_t$ depends only on the previous state $\boldsymbol{\Theta}_{t-1}$ and has nothing to do with earlier states. Therefore, we can denote the transition probability of the forward process as $q(\boldsymbol{\Theta}_t|\boldsymbol{\Theta}_{t-1})$. According to Bayes' formula, we can derive $q(\boldsymbol{\Theta}_{t-1}|\boldsymbol{\Theta}_t, \boldsymbol{\Theta}_0)$ as follows:

$$q(\boldsymbol{\Theta}_{t-1}|\boldsymbol{\Theta}_t, \boldsymbol{\Theta}_0) = q(\boldsymbol{\Theta}_t|\boldsymbol{\Theta}_{t-1})\frac{q(\boldsymbol{\Theta}_{t-1}|\boldsymbol{\Theta}_0)}{q(\boldsymbol{\Theta}_t|\boldsymbol{\Theta}_0)} \tag{17}$$

Diffusion models often learn data distributions efficiently by minimizing a variational upper bound. Specifically, the variational upper bound is formally defined as follows:

$$\begin{aligned}\mathcal{L}_{\text{vb}}(\boldsymbol{\Theta}_0) := \mathbb{E}_{q(\boldsymbol{\Theta}_0)}\Bigg[\underbrace{D_{KL}(q(\boldsymbol{\Theta}_T|\boldsymbol{\Theta}_0) \parallel p_\theta(\boldsymbol{\Theta}_T))}_{\mathcal{L}_T}\Bigg] \\ + \sum_{t=1}^{T}\mathbb{E}_{q(\boldsymbol{\Theta}_t|\boldsymbol{\Theta}_0)}\underbrace{D_{KL}(q(\boldsymbol{\Theta}_{t-1}|\boldsymbol{\Theta}_t, \boldsymbol{\Theta}_0) \parallel p_\theta(\boldsymbol{\Theta}_{t-1}|\boldsymbol{\Theta}_t))}_{\mathcal{L}_t} \\ \underbrace{-\mathbb{E}_{q(\boldsymbol{\Theta}_1|\boldsymbol{\Theta}_0)}\log(p_\theta(\boldsymbol{\Theta}_0|\boldsymbol{\Theta}_1))}_{\mathcal{L}_0}\end{aligned} \tag{18}$$

Many DDPMs implementations have found that using alternative loss functions can bring significant benefits. For example, [20] derived a simplified loss function that simplifies loss calculation by reweighting the terms of ELBO, which can effectively reduce the complexity of the training process while maintaining the generation quality of the model. [1,2] proposed the use of hybrid loss methods, which balances the performance of the model in different tasks by combining different types of loss functions. DDSS replaces the original KL term $\mathcal{L}_t$ by parameterizing $p_\theta(\Theta_0|\Theta_t)$. Specifically, $\mathcal{L}_t$ is replaced by $\mathcal{L}_t = -\log(p_\theta(\Theta_0|\Theta_t))$. In addition, the reweighting term $1 - 2\cdot\overline{\beta}_t + \frac{1}{T}$ assigns more linear importance to samples with less noise, which can learn the essential characteristics when processing samples with less noise. The simplified loss definition is as follows:

$$\mathcal{L}_{\text{s}} = -\mathbb{E}_{q(\boldsymbol{\Theta}_0)}\sum_{t=1}^{T}\left(1 - 2\cdot\overline{\beta}_t + \frac{1}{T}\right)\cdot\mathbb{E}_{q(\boldsymbol{\Theta}_t|\boldsymbol{\Theta}_0)}\log p_\theta\left(\boldsymbol{\Theta}_0 \mid \boldsymbol{\Theta}_t\right) \tag{19}$$

3.2. Score-based generative models (SGMs)

SGMs [19,31] use score matching techniques to generate high-quality data. The core idea of SGM is to estimate the score function of data distribution through score matching techniques. The score function can reveal the trend and direction of data changes at specific points, effectively guiding the generation process and gradually generating samples that conform to the data distribution from noise [113].

First, SGMs considers the forward direction of the diffusion process, i.e., starting from the initial state $m(0) \sim p_0$, the system gradually evolves to $m(T) \sim p_T$ over time [142]. However, according to Anderson's research, if we start from the final state $m(T) \sim p_T$ and evolve through the reverse time process, we can successfully trace back to the initial state $m(0) \sim p_0$ of the system. This means that its time is backward. Specifically, this reverse diffusion process can be described by a reverse-time SDE, which enables us to describe how the system evolves from $m(T) \sim p_T$ to $m(0) \sim p_0$ in reverse:

$$\mathrm{d}\boldsymbol{m} = [\boldsymbol{f}(\boldsymbol{m}, t) - g(t)^2\nabla_{\boldsymbol{m}}\log p_t(\boldsymbol{m})]\mathrm{d}t + g(t)\mathrm{d}\bar{\boldsymbol{w}} \tag{20}$$

In the reverse time process, the Wiener process w describes the random behavior of the system in the reverse time direction, which is similar to the standard Brownian motion, except that time flows backwards. The negative time step dt represents the small step forward in time, thus ensuring that we can gradually reverse the evolution of the system. After knowing the score function $\nabla_m \log p_t(m)$ at each time point t, we can reconstruct the early state of the system through the inverse diffusion equation. The score function provides us with information on how to adjust m at each time point, making the reverse simulation feasible.

The core idea of score matching is to approximate the gradient of the data distribution through a network [32,107]. To estimate $\nabla_m \log p_t(m)$, SGMs generalize the score matching method to the time domain and introduce a time-dependent score-based network, which can capture the dynamic changes of data distribution in time. Specifically, the optimization objective is defined as follows:

$$\begin{aligned}\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi}}\mathbb{E}_t\Big\{\gamma(t)\mathbb{E}_{\boldsymbol{m}(0)}\mathbb{E}_{\boldsymbol{m}(t)|\boldsymbol{m}(0)}\Big[\parallel \boldsymbol{s}_{\boldsymbol{\phi}}(\boldsymbol{m}(t), t) \\ -\nabla_{\boldsymbol{m}(t)}\log p_{0t}(\boldsymbol{m}(t) \mid \boldsymbol{m}(0)) \parallel_2^2\Big]\Big\}\end{aligned} \tag{21}$$

where $\gamma : [0, T] \mapsto \mathbb{R}_{>0}$ is a positive weight function, which can balance the score functions at different time steps during training, allowing to learn the dynamic changes of data distribution more stably. The data point $m(t)$ at time t comes from the conditional distribution $p_{0t}(\boldsymbol{m}(t) \mid \boldsymbol{m}(0))$.

SGMs follows the same paradigm as image generation in graph generation tasks. First, the distribution of the graph data to be generated is clarified, and the goal is to generate new graphs from this data distribution [19,32,107]. To utilize the score matching method, the original graph data is added with different degrees of noise to construct a series of noisy graphs. Next, a scoring network is trained to estimate the gradient of the graph data at each noise level. The goal of the scoring function is to guide how to restore the original graph from the noisy graph. Once the scoring network is trained, it can be restored to the form of the original graph by gradually denoising. Next we summarize several representative works on graph generation using SGMs.

3.2.1. Graph diffusion via SDE systems (GDSS)

GDSS [32] proposes a new score-based graph generation model with a continuous time framework, which successfully models the joint distribution of graph via the graph diffusion process and SDE system. The customized score matching target and the new SDE solver enable the model to effectively sample and generate new graphs during the reverse diffusion process. The results show that GDSS not only generates graphs close to the training distribution, but also maintains the chemical valence rules in the chemical molecule generation task, demonstrating its potential and effectiveness in modeling complex graph data. As shown in Fig. 3, GDSS (green) can effectively capture the potential relationship between features and structures through its diffusion process by modeling dependencies between components, thereby generating new samples. GDSS-seq (red) adopts a step-by-step generation method, that is, generating features $\boldsymbol{\mathcal{X}}$ and adjacency matrix $\boldsymbol{\Theta}$ in sequence. The EDP-GNN [107] framework only generates the adjacency matrix Θ, and the node features $\boldsymbol{\mathcal{X}}$ are directly sampled from the training data. Different from the continuous-time diffusion process of GDSS and GDSS-seq, the node features in EDP-GNN are statically sampled.

Forward Graph Diffusion. Forward Graph Diffusion is a process of describing the propagation of information in a graph by using a SDE system defined on the graph structure [117]. The diffusion process can capture the complex relationships and topological structures. Specifically, forward graph diffusion can be represented by the following SDE system:

$$\begin{aligned}\mathrm{d}\boldsymbol{\mathcal{X}}_t &= \boldsymbol{f}^{\mathcal{X}}(\boldsymbol{\mathcal{X}}_t, t)\mathrm{d}t + \sigma_{\mathcal{X},t}\mathrm{d}\boldsymbol{\mathcal{B}}_t^{\mathcal{X}} \\ \mathrm{d}\boldsymbol{\Theta}_t &= \boldsymbol{f}^{A}(\boldsymbol{\Theta}_t, t)\mathrm{d}t + \sigma_{\Theta,t}\mathrm{d}\boldsymbol{\mathcal{B}}_t^{\Theta}\end{aligned} \tag{22}$$

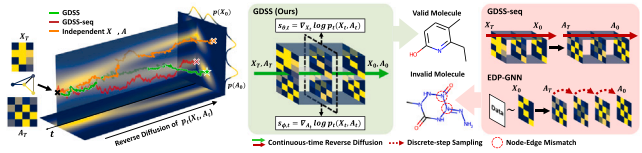

Fig. 3. Schematic diagram of the overall process of GDSS. The forward diffusion process involves adding noise to the graph data in continuous time, effectively perturbing the structure and node features of the graph. The reverse process reconstructs the original graph data by gradually removing the noise [32].

where $\boldsymbol{\mathcal{X}}_t$ is the feature representation at time t. $\boldsymbol{f}^{\mathcal{X}}(\boldsymbol{\mathcal{X}}_t, t) : \mathbb{R}^{n\times n} \mapsto \mathbb{R}^{n\times n}$ is the drift term, describing the change of the nodes features and adjacency matrix. The drift term can include the feature update of the node itself and the interaction with neighboring nodes. $\sigma_{\mathcal{X},t}, \sigma_{\Theta,t}$ is the diffusion coefficient, describing the random fluctuation of the feature. $\boldsymbol{B}_t^{\mathcal{X}}, \boldsymbol{B}_t^{\Theta}$ is a standard Brownian motion, representing a random perturbation.

Reversed Graph Diffusion. The stochastic differential equations for reverse time graph diffusion are used to describe the diffusion process of information in graph data in the reverse time direction [32]. The diffusion process usually involves deterministic changes and random perturbations of node features to capture the complex relationships. Specifically, the stochastic differential equations for reverse time graph diffusion can be expressed as:

$$\begin{aligned} \mathrm{d}\bar{\boldsymbol{\mathcal{X}}}_t &= \left(\boldsymbol{f}^{\mathcal{X}}(\bar{\boldsymbol{\mathcal{X}}}_t, t) - \sigma_{\mathcal{X},t}^2 \nabla_{\boldsymbol{\mathcal{X}}} \log p_t(\bar{\boldsymbol{\mathcal{X}}}_t, \bar{\boldsymbol{\Theta}}_t)\right) \mathrm{d}\bar{t} + \sigma_{\mathcal{X},t} \mathrm{d}\bar{\boldsymbol{B}}_t^{\boldsymbol{\mathcal{X}}} \\ \mathrm{d}\bar{\boldsymbol{\Theta}}_t &= \left(\boldsymbol{f}^{\boldsymbol{\Theta}}(\bar{\boldsymbol{\Theta}}_t, t) - \sigma_{\Theta,t}^2 \nabla_{\boldsymbol{\Theta}} \log p_t(\bar{\boldsymbol{\mathcal{X}}}_t, \bar{\boldsymbol{\Theta}}_t)\right) \mathrm{d}\bar{t} + \sigma_{\Theta,t} \mathrm{d}\bar{\boldsymbol{B}}_t^{\boldsymbol{\Theta}} \end{aligned} \tag{23}$$

$\boldsymbol{\mathcal{X}}_t \perp \boldsymbol{\Theta}_t | \mathcal{G}_0$ means that given the initial condition $\mathcal{G}_0$, the state variables $\boldsymbol{\mathcal{X}}_t$ and $\boldsymbol{\Theta}_t$ are independent of each other. Therefore, the joint probability density function $p_{t|0}(\hat{\boldsymbol{\mathcal{X}}}_t, \hat{\boldsymbol{\Theta}}_t)$ can be decomposed into two independent parts: one is the part $p_{t|0}(\boldsymbol{\mathcal{X}}_t|\boldsymbol{\mathcal{X}}_0)$ related to $\boldsymbol{\mathcal{X}}_t$ and the initial state $\boldsymbol{\mathcal{X}}_0$, and the other is the part $p_{t|0}(\boldsymbol{\Theta}_t|\boldsymbol{\Theta}_0)$ related to $\boldsymbol{\Theta}_0$ and the initial state $\boldsymbol{\Theta}_0$. This decomposition form greatly simplifies the complex calculation and estimation of the joint probability density function $\mathcal{G}_t|\mathcal{G}_0$ in the denoising score matching objective function and reduces the computational complexity. The simplified form is defined as follows:

$$\begin{aligned} \hat{\mathcal{E}}(\boldsymbol{\theta}) &\triangleq \mathbb{E}_{\mathcal{G}\sim\mathrm{Unif}(S)} \mathbb{E}_{\mathcal{G}_t|\mathcal{G}} \parallel s_{\boldsymbol{\theta}}(\mathcal{G}_t) - \nabla \log p_{t|0}(\boldsymbol{\mathcal{X}}_t|\boldsymbol{\mathcal{X}}_0) \parallel^2 \\ \hat{\mathcal{E}}(\boldsymbol{\phi}) &\triangleq \mathbb{E}_{\mathcal{G}\sim\mathrm{Unif}(S)} \mathbb{E}_{\mathcal{G}_t|\mathcal{G}} \parallel s_{\boldsymbol{\phi}}(\mathcal{G}_t) - \nabla \log p_{t|0}(\boldsymbol{\Theta}_t|\boldsymbol{\Theta}_0) \parallel^2 \end{aligned} \tag{24}$$

The expected value in Eq. (24) can be computed by using the Monte Carlo estimation. It should be noted that estimating the partial score is not equivalent to estimating $\nabla_{\boldsymbol{\mathcal{X}}_t} \log p_t(\boldsymbol{\mathcal{X}}_t)$ or $\nabla_{\boldsymbol{\Theta}_t} \log p_t(\boldsymbol{\Theta}_t)$. Specifically, estimating the partial score requires considering the joint distribution of $\boldsymbol{\mathcal{X}}_t$ and $\boldsymbol{\Theta}_t$ at time t, rather than just considering the marginal distribution of one of the variables separately. The score function of this joint distribution is more complicated because it involves the dynamic relationship between the two variables rather than the static properties of a single variable. By using the objective of Eq. (24), the remaining task is to find a partial score model that learns the underlying distribution of the graph. The model needs to be flexible and expressive enough to capture the complex relationship between $\boldsymbol{\mathcal{X}}_t$ and $\boldsymbol{\Theta}_t$ at time t, while also being able to accurately estimate the score function of the joint distribution at different points in time. Therefore, GDSS proposes a new score-based model architecture. This new architecture aims to improve the expressiveness of the model, enabling it to better learn and estimate partial scores of the underlying distribution of the graph, thereby achieving more accurate sampling and inference.

Algorithm 1 Optimizing GDSS

Input: $\boldsymbol{s}_{\boldsymbol{\phi},t}(\cdot)$, $\boldsymbol{f}^{\mathcal{X}}(\cdot,t)$, $\boldsymbol{f}^{\Lambda}(\cdot,t)$, $\sigma_{X,t}, \sigma_{\Lambda,t}$.
Initialize $\boldsymbol{\theta}_0, \boldsymbol{\phi}_0$
$(\boldsymbol{\mathcal{X}}_0, \boldsymbol{\Theta}_0) \sim \mathcal{G}$
$t \sim \mathrm{Unif}([0, T])$
$\boldsymbol{\mathcal{X}}_t \sim \int_0^t \boldsymbol{f}^{X}(\boldsymbol{\mathcal{X}}_\tau, \tau)\mathrm{d}\tau + \int_0^t \sigma_{\mathcal{X},\tau} \mathrm{d}\boldsymbol{B}_\tau^{\mathcal{X}}$,
$\boldsymbol{\Theta}_t \sim \int_0^t \boldsymbol{f}^{\Theta}(\boldsymbol{\mathcal{X}}_\tau, \tau)\mathrm{d}\tau + \int_0^t \sigma_{\Theta,\tau} \mathrm{d}\boldsymbol{B}_\tau^{\Theta}$
$\boldsymbol{s}_{\boldsymbol{\phi},t}(\mathcal{G}_t) = \mathrm{MLP}\left(\left[\{\mathrm{GMH}\left(\boldsymbol{H}_i, \boldsymbol{\Theta}_t^p\right)\}_{i=0,p=1}^{K,P}\right]\right)$
$s_{\theta,t}(\mathcal{G}_t) = \mathrm{MLP}([\{\boldsymbol{H}_i\}_{i=0}^{L}])$
$\hat{\mathcal{E}}(\boldsymbol{\theta}) \triangleq \mathbb{E}_{\mathcal{G}\sim\mathrm{Unif}(S)} \mathbb{E}_{\mathcal{G}_t|\mathcal{G}} \| s_{\boldsymbol{\theta}}(\mathcal{G}_t) - \nabla \log p_{t|0}(\boldsymbol{\mathcal{X}}_t|\boldsymbol{\mathcal{X}}_0) \|^2$
$\hat{\mathcal{E}}(\boldsymbol{\phi}) \triangleq \mathbb{E}_{\mathcal{G}\sim\mathrm{Unif}(S)} \mathbb{E}_{\boldsymbol{G}_t|\mathcal{G}} \| s_{\boldsymbol{\phi}}(\mathcal{G}_t) - \nabla \log p_{t|0}(\boldsymbol{\Theta}_t|\boldsymbol{A}_0) \|^2$
Return: $\boldsymbol{\theta}_K, \boldsymbol{\phi}_K$

Algorithm 2 Sampling from GDSS

$t = T$
Sample $\boldsymbol{\mathcal{X}}_K, \boldsymbol{\Theta}_K \sim p_T$
for $i = K - 1$ **to** 0 **do**
  $\boldsymbol{S}_{\mathcal{X}} \leftarrow \boldsymbol{s}_{\theta,t}(\boldsymbol{\mathcal{X}}_{i+1}, \boldsymbol{\Theta}_{i+1})$
  $\boldsymbol{S}_{\Theta} \leftarrow \boldsymbol{s}_{\phi,t}(\boldsymbol{\mathcal{X}}_{i+1}, \boldsymbol{\Theta}_{i+1})$
  $\boldsymbol{\mathcal{X}}_{i+1} \leftarrow \boldsymbol{\mathcal{X}}_{i+1} + \frac{\alpha}{2}\boldsymbol{S}_{\mathcal{X}} + \epsilon_s\sqrt{\alpha}\boldsymbol{z}_{\mathcal{X}}$
  $\boldsymbol{\Theta}_{i+1} \leftarrow \boldsymbol{\Theta}_{i+1} + \frac{\alpha}{2}\boldsymbol{S}_{\Theta} + \epsilon_s\sqrt{\alpha}\boldsymbol{z}_{A}$
  $t' \leftarrow t - \delta t/2$
  $\tilde{\boldsymbol{\mathcal{X}}}_i \sim p_{t,t'}(\tilde{\boldsymbol{\mathcal{X}}}_i | \boldsymbol{\mathcal{X}}_{i+1})$
  $\tilde{\boldsymbol{\Theta}}_i \sim p_{t,t'}(\tilde{\boldsymbol{A}}_i | \boldsymbol{\Theta}_{i+1})$
  $\tilde{\boldsymbol{\mathcal{X}}}_i \leftarrow \tilde{\boldsymbol{\mathcal{X}}}_i + g_{1,t}^2 \boldsymbol{S}_{\mathcal{X}} \delta t$
  $\tilde{\boldsymbol{\Theta}}_i \leftarrow \tilde{\boldsymbol{\Theta}}_i + g_{2,t}^2 \boldsymbol{S}_{\Theta} \delta t$
  $t \leftarrow t - \delta t$
  $\boldsymbol{\mathcal{X}}_i \sim p_{t',t}(\boldsymbol{\mathcal{X}}_i | \tilde{\boldsymbol{\mathcal{X}}}_i)$
  $\boldsymbol{\Theta}_i \sim p_{t',t}(\boldsymbol{\Theta}_i | \tilde{\boldsymbol{\Theta}}_i)$
end for
Return: $\boldsymbol{\mathcal{X}}_0$, $\boldsymbol{\Theta}_0$

Specifically, GDSS proposes a new time-based scoring model architecture based on the graph attention network [143–148], which can capture the time-varying dependencies between $\boldsymbol{\mathcal{X}}_t$ and $\boldsymbol{\Theta}_t$ and distinguish the important relationships between nodes. In addition, GDSS further uses high-order adjacency matrices to represent long-distance dependencies. High-order adjacency matrices can capture the mutual influence between distant nodes in the graph structure, which is crucial for modeling complex temporal dependencies. Specifically, GDSS first constructs the score-based modulus $s_{\phi,t}$ to estimate $\nabla_{\boldsymbol{\Theta}_t} \log p_t(\boldsymbol{\mathcal{X}}_t, \boldsymbol{\Theta}_t)$ with the same dimension as $\boldsymbol{\Theta}_t$ as follows:

$$\boldsymbol{s}_{\boldsymbol{\phi},t}(\mathcal{G}_t) = \mathrm{MLP}\left(\left[\{\mathrm{GMH}\left(\boldsymbol{H}_i, \boldsymbol{\Theta}_t^p\right)\}_{i=0,p=1}^{K,P}\right]\right) \tag{25}$$

where $\boldsymbol{H}_0 = \boldsymbol{\mathcal{X}}_t$, $\boldsymbol{H}_{i+1} = \mathrm{GHM}(\boldsymbol{\Theta}_t, \boldsymbol{H}_i)$, GHM is a GCN layer.

GDSS also uses MLP to improve the linear representation ability of node features as:

$$s_{\theta,t}(\mathcal{G}_t) = \mathrm{MLP}([\{\boldsymbol{H}_i\}_{i=0}^{L}]) \tag{26}$$

Next, GDSS uses the trained score models $\boldsymbol{s}_{\phi,t}$ and $\boldsymbol{s}_{\theta,t}$ to estimate the scores of $\boldsymbol{\mathcal{X}}_t$ and $\boldsymbol{\Theta}_t$, respectively. By applying these two score models to the reverse time SDE system, GDSS can gradually approximate the inverse diffusion process. Specifically, at each time step, GDSS uses $\boldsymbol{s}_{\theta,t}$ and $\boldsymbol{s}_{\phi,t}$ to calculate the score of the current state and updates the state according to the reverse time SDE as:

$$\begin{cases} \mathrm{d}\boldsymbol{\mathcal{X}}_t = \boldsymbol{f}_{1,t}(\boldsymbol{\mathcal{X}}_t)\mathrm{d}\bar{t} + g_{1,t}\mathrm{d}\bar{\boldsymbol{w}}_1 - g_{1,t}^2 \boldsymbol{s}_{\theta,t}(\boldsymbol{\mathcal{X}}_t, \boldsymbol{\Theta}_t)\mathrm{d}\bar{t} \\ \mathrm{d}\boldsymbol{\Theta}_t = \underbrace{\boldsymbol{f}_{2,t}(\boldsymbol{\Theta}_t)\mathrm{d}\bar{t} + g_{2,t}\mathrm{d}\bar{\boldsymbol{w}}_2}_{F} \underbrace{-g_{2,t}^2 \boldsymbol{s}_{\phi,t}(\boldsymbol{\mathcal{X}}_t, \boldsymbol{\Theta}_t)\mathrm{d}\bar{t}}_{S} \end{cases} \tag{27}$$

From the result of the symmetric Trotter theorem [149], it can be seen that the propagation of the calibration state $\mathcal{G}'_t$ from time t to $t - \delta t/2$ follows the dynamics of Eq. (27) Specifically, Eq. (27) can be approximated by applying $e^{\frac{\delta t}{2}\hat{\mathcal{L}}^*_F} e^{\delta t \hat{\mathcal{L}}^*_S} e^{\frac{\delta t}{2}\hat{\mathcal{L}}^*_F}$ to $\mathcal{G}'_t$ as:

$$e^{\frac{\delta t}{2}\hat{\mathcal{L}}^*_F}\mathcal{G} = \tilde{\mathcal{G}} \sim p_{t,t-\delta t/2}(\tilde{\mathcal{G}}|\mathcal{G}) \tag{28}$$

The pseudo codes of GDSS optimization and sampling process are shown in Algorithm 1 and Algorithm 2.

3.2.2. Graph spectral diffusion model (GSDM)

GSDM [125] argues that running full-rank diffusion SDE on the entire adjacency matrix space involves a lot of computing resources and time. Specifically, the adjacency matrix is usually sparse, but full-rank diffusion SDE requires processing the entire matrix, which leads to a greatly increased computational burden, especially when dealing with large-scale graph data. In addition, full-rank diffusion SDE introduces a lot of noise, which causes unnecessary perturbations to the topological structure of the graph, making it difficult to accurately capture the true topological structure of the graph and resulting in a decrease in the quality of the generated data.

Algorithm 3 Optimizing GSDM

Input: $s_{\phi,t}(\cdot)$, $f^{\mathcal{X}}(\cdot,t)$, $f^{\Lambda}(\cdot,t)$, $\sigma_{\mathcal{X},t}, \sigma_{\Lambda,t}$.
Initialize θ_0, ϕ_0
$(\mathcal{X}_0, \Theta_0) \sim \mathcal{G}$
$\Lambda_0 \leftarrow \text{EigenValues}(\Theta_0)$
$t \sim \text{Unif}([0,T])$
$\mathcal{X}_t \sim \int_0^t f^{\mathcal{X}}(\mathcal{X}_\tau,\tau)\mathrm{d}\tau + \int_0^t \sigma_{\mathcal{X},\tau}\mathrm{d}\mathcal{B}^{\mathcal{X}}_\tau$,
$\Lambda_t \sim \int_0^t f^{\Lambda}(\Lambda_\tau,\tau)\mathrm{d}\tau + \int_0^t \sigma_{\Lambda,\tau}\mathrm{d}\mathcal{B}^{\Lambda}_\tau$
$\widehat{\mathcal{E}}(\theta_k) \leftarrow \|s_{\theta_k}(\mathcal{X}_t,\Lambda_t) - \nabla \log p_{t|0}(\mathcal{X}_t|\mathcal{X}_0)\|^2$
$\widehat{\mathcal{E}}(\phi_k) \leftarrow \|s_{\phi_k}(\mathcal{X}_t,\Lambda_t) - \nabla \log p_{t|0}(\Lambda_t|\Lambda_0)\|^2$
Return: θ_K, ϕ_K

Algorithm 4 Sampling from GSDM

$t \leftarrow T$
$(\hat{\mathcal{X}}_T, \hat{\Lambda}_T) \sim \pi$, $\hat{U}_0 \sim \text{Unif}(\{U \triangleq \text{EigenVectors}(\Theta), (\mathcal{X},\Theta) \sim \text{Data}\})$
for $m = M-1$ **to** 0 **do**
  $S_{\mathcal{X}} \leftarrow s_{\theta,t}(\hat{\mathcal{X}}_t, \hat{\Lambda}_t, \hat{U}_0)$, $S_{\Lambda} \leftarrow s_{\phi,t}(\hat{\mathcal{X}}_t, \hat{\Lambda}_t, \hat{U}_0)$
  $t' \leftarrow t - T/(2M)$
  $\hat{\mathcal{X}}_{t'} \leftarrow (2-\sqrt{1-\beta_{m+1}})\hat{\mathcal{X}}_t + \beta_{m+1}S_{\mathcal{X}}) + \sqrt{\beta_{m+1}}z_{\mathcal{X}}$
  $z_{\mathcal{X}} \sim N(\mathbf{0}, \mathbf{I})$
  $\hat{\Lambda}_{t'} \leftarrow (2-\sqrt{1-\beta_{m+1}})\hat{\Lambda}_t + \beta_{m+1}S_{\Lambda}) + \sqrt{\beta_{m+1}}z_{\Lambda}$,
  $z_{\Lambda} \sim N(\mathbf{0}, \mathbf{I})$
  $S_{\mathcal{X}} \leftarrow s_{\theta,t'}(\hat{X}_{t'}, \hat{\Lambda}_{t'}, \hat{U}_0)$, $S_{\Lambda} \leftarrow s_{\phi,t'}(\hat{\mathcal{X}}_{t'}, \hat{\Lambda}_{t'}, \hat{U}_0)$
  $t \leftarrow t' - T/(2M)$
  $\hat{\mathcal{X}}_t \leftarrow \hat{\mathcal{X}}_{t'} + \epsilon_i S_{\mathcal{X}} + \sqrt{2\epsilon_i}z_{\mathcal{X}}$
  $\hat{\Lambda}_t \leftarrow \hat{\Lambda}_{t'} + \epsilon_i S_{\Lambda} + \sqrt{2\epsilon_i}z_{\Lambda}$
end for
$\hat{\Theta}_0 = \hat{U}_0\hat{\Lambda}_0\hat{U}_0^T$
Return: $(\hat{\mathcal{X}}_0, \hat{\Theta}_0)$

Therefore, GSDM designs a novel Graph Spectral Diffusion Model. The core idea of the GSDM is to capture graph structure information through feature decomposition of the graph, and to propagate node features or label information through the diffusion process. Specifically, graph eigendecomposition is to perform eigenvalue decomposition on the Laplacian matrix to obtain eigenvalues and eigenvectors. For a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the node set, E is the edge set, the Laplacian matrix L is defined as $L = D - \Theta$. Perform eigenvalue decomposition of the Laplacian matrix L to obtain $L = U\Lambda U^T$, where U is the eigenvector matrix and Λ is the diagonal eigenvalue matrix. Through the diffusion process, node features can be smoothed in the graph. Using the eigenvector U and eigenvalue Λ of the graph, we can define the forward spectral diffusion via SDE system as follows:

$$\begin{aligned}\mathrm{d}\mathcal{X}_t &= f^{\mathcal{X}}(\mathcal{X}_t,t)\mathrm{d}t + \sigma_{\mathcal{X},t}\mathrm{d}\mathcal{B}^{\mathcal{X}}_t,\\ \mathrm{d}\Lambda_t &= f^{\Lambda}(\Lambda_t,t)\mathrm{d}t + \sigma_{\Lambda,t}\mathrm{d}W^{\Lambda}_t\end{aligned} \tag{29}$$

where $t \in [0,1]$ is the time parameter, $(\mathcal{X}_0, A_0) \sim \mathcal{G}, \Theta_0 = U_0\Lambda_0U_0^\top$, $f^{\mathcal{X}}(\cdot,t)$ is the drift function, $f^{\Lambda}(\cdot,t)$ is the drift function for spectrum. $\sigma_{\mathcal{X},t}, \sigma_{\Lambda,t}$ is the diffusion coefficient, describing the random fluctuation of the feature. $\mathcal{B}^{\mathcal{X}}_t, \mathcal{B}^{A}_t$ is a standard Brownian motion or Wiener process, representing a random perturbation, and $W^{\Lambda}_t \triangleq \text{diag}(\mathcal{B}^{\Lambda}_t)$ is a diagonal Brownian motion.

By defining the evolution process of $\Theta_t \triangleq U_0\Lambda_tU_0^\top$ and using the n-dimensional Gaussian process to drive the change of eigenvalues, we can effectively prevent the adjacency matrix from spreading arbitrarily in the entire space. On this basis, the inverse time spectral diffusion stochastic differential equation is established, which can gradually reverse the damage process of the adjacency matrix and generate high-quality graph data similar to the original graph. Specifically, the reversed time spectral diffusion SDE system is defined as follows:

$$\begin{cases}\mathrm{d}\bar{\mathcal{X}}_t = \left(f^{\mathcal{X}}(\bar{\mathcal{X}}_t,t) - \sigma^2_{\mathcal{X},t}\nabla_{\mathcal{X}}\log p_t(\bar{\mathcal{X}}_t,\bar{\Lambda}_t)\right)\mathrm{d}\bar{t} + \sigma_{\mathcal{X},t}\mathrm{d}\bar{\mathcal{B}}^{\mathcal{X}}_t\\ \mathrm{d}\bar{\Lambda}_t = \left(f^{\Lambda}(\bar{\Lambda}_t,t) - \sigma^2_{\Lambda,t}\nabla_{\Lambda}\log p_t(\bar{\mathcal{X}}_t,\bar{\Lambda}_t)\right)\mathrm{d}\bar{t} + \sigma_{\Lambda,t}\mathrm{d}\bar{W}^{\Lambda}_t\end{cases} \tag{30}$$

where $\mathrm{d}\bar{t} = -\mathrm{d}t$ is the negative in finitesimal time step.

Boundary conditions are imposed on the joint distribution of $(\mathcal{X}_1, \Lambda_1)$ so that the joint distribution of the node feature ($\mathcal{X}_1$ and the eigenvalue matrix Λ_1) at the initial time $t = 1$ is the same as the prior distribution π. GSDM can use the inverse time spectrum diffusion SDE to recover the true distribution $(\mathcal{X}_0, \Lambda_0)$ from the prior distribution π. According to the score matching technique, the two score networks $s_\theta(\cdot,\cdot,\cdot)$ and $s_\phi(\cdot,\cdot,\cdot)$ are trained by minimizing the loss function so that they approximate the two score functions $\nabla_{\mathcal{X}}\log p_t(\mathcal{X},\Lambda)$ and $\nabla_{\Lambda}\log p_t(\mathcal{X},\Lambda)$ respectively as follows:

$$\begin{aligned}\widehat{\mathcal{E}}(\theta) &\triangleq \mathbb{E}_{\mathcal{G}\sim\text{Unif}(S)}\mathbb{E}_{\mathcal{X}_t|\mathcal{G}} \parallel s_\theta(\mathcal{X}_t,\Lambda_t,U_0) - \nabla\log p_{t|0}(\mathcal{X}_t|\mathcal{X}_0) \parallel^2\\ \widehat{\mathcal{E}}(\phi) &\triangleq \mathbb{E}_{\mathcal{G}\sim\text{Unif}(S)}\mathbb{E}_{\Lambda_t|\mathcal{G}} \parallel s_\phi(\mathcal{X}_t,\Lambda_t,U_0) - \nabla\log p_{t|0}(\Lambda_t|\Lambda_0) \parallel^2\end{aligned} \tag{31}$$

The pseudo codes of GDSS optimization and sampling process are shown in Algorithm 3 and Algorithm 4.

3.3. Stochastic differential equations (SDEs)

SDEs [32] model implements the forward and backward diffusion processes through a continuous-time stochastic process. The SDE model describes the diffusion process as a continuous stochastic process rather than discrete time steps [150,151]. Specifically, a data point is usually gradually diffused from the data space to the Gaussian noise space through SDEs [117].

Forward Process: The forward process perturbs the data into noise through SDE [60]. The equation is defined as follows:

$$d\mathcal{X}_t = f(\mathcal{X}_t,t)dt + g(t)dW_t \tag{32}$$

Reverse Process: The reverse process converts noise into data through reverse time SDE [117]. This process is the reverse operation of the forward process, removing the gradually added data noise and reconstructing a clear image or data. The equation is defined as follows:

$$d\mathcal{X}_t = \left[f(\mathcal{X}_t,t) - g(t)^2\nabla_x\log p_t(\mathcal{X}_t)\right]dt + g(t)d\tilde{W}_t \tag{33}$$

where $\nabla_x \log p_t(\mathcal{X}_t)$ is the gradient information of the data, which helps to guide the reverse process.

Since continuous SDEs are difficult to handle in practical computations, discretization schemes (e.g., Euler–Maruyama) are used to discretize the SDE into a series of time-step update processes and used

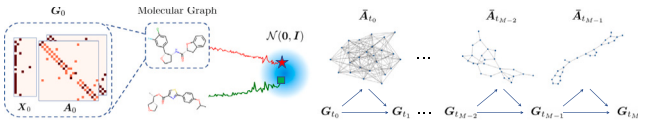

Fig. 4. An example illustrating the overall CDGS process [120].

to approximately solve the SDE. The discretization process is defined as follows:

$$\mathcal{X}_{t+\Delta t} = \mathcal{X}_t + f(\mathcal{X}_t, t)\Delta t + g(t)\sqrt{\Delta t}Z \tag{34}$$

The application of SDEs in graph generation provides an effective method to generate the target graph structure from initial noise by defining the diffusion process of nodes and edges [129,142]. Thanks to the development of SDEs and diffusion models, graph generation tasks have made significant progress in many fields and have shown broad application prospects [33,113,152–156]. Next we summarize several representative works on graph generation using SDEs.

3.3.1. Conditional diffusion graph structures (CDGS)

Understanding the underlying distribution requires not only accurately modeling the complex distribution of molecular structures, but also the ability to quickly generate new molecular graphs that meet practical needs. To address this challenge, CDGS [120] proposes a conditional diffusion model to generate molecular graphs. As shown in Fig. 4, CDGS constructs a forward graph diffusion process and intrinsic features with SDEs, and derives the discrete graph structure as a condition for the reverse generation process.

Specifically, given a graph $\mathcal{G} = (\boldsymbol{\Theta}, \boldsymbol{\mathcal{X}})$, the node feature matrix $\boldsymbol{\mathcal{X}}$ is represented by an F-dimensional real vector. The edge information matrix $\boldsymbol{\Theta}$ describes the edge connections and types in the graph. The forward diffusion process of the graph can be described by SDE, and the time variable $t \in [0, T]$ represents the time scale of the diffusion process, starting from the initial state $t = 0$ and ending at the final state $t = T$ as follows:

$$\mathrm{d}\mathcal{G}_t = f(t)\mathcal{G}_t\mathrm{d}t + g(t)\mathrm{d}\boldsymbol{w}_t \tag{35}$$

where $f(t) = \frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}$ denotes the drift coefficient, $g^2(t) = \frac{\mathrm{d}\sigma_t^2}{\mathrm{d}t} - 2\frac{\mathrm{d}\log\alpha_t}{\mathrm{d}t}\sigma_t^2$ is the diffusion coefficient. Correspondingly, the reverse-time SDE is formally defined as:

$$\mathrm{d}\mathcal{G}_t = [f(t)\mathcal{G}_t - g^2(t)\nabla_{\mathcal{G}}\log q_t(\mathcal{G}_t)]\mathrm{d}_t + g(t)\mathrm{d}\bar{\boldsymbol{w}}_t \tag{36}$$

To simplify the above steps, CDGS splits it into two parts that share the same drift coefficient and diffusion coefficient:

$$\begin{cases} \mathrm{d}\boldsymbol{\mathcal{X}}_t = [f(t)\boldsymbol{\mathcal{X}}_t - g^2(t)\nabla_{\boldsymbol{\mathcal{X}}}\log q_t(\boldsymbol{\mathcal{X}}_t, \boldsymbol{\Theta}_t)]\mathrm{d}_t + g(t)\mathrm{d}\bar{\boldsymbol{w}}_t^1 \\ \mathrm{d}\boldsymbol{\Theta}_t = [f(t)\boldsymbol{\Theta}_t - g^2(t)\nabla_{\boldsymbol{\Theta}}\log q_t(\boldsymbol{\mathcal{X}}_t, \boldsymbol{\Theta}_t)]\mathrm{d}_t + g(t)\mathrm{d}\bar{\boldsymbol{w}}_t^2 \end{cases} \tag{37}$$

CDGS uses a network $\epsilon_\theta(\mathcal{G}_t, \bar{\boldsymbol{\Theta}}_t, t)$ with discrete graph structure conditions to parameterize the score. Specifically, the neural network ϵ_θ accepts the graph $\mathcal{G}_t$ and its discrete graph structure $\bar{\boldsymbol{\Theta}}$ and time t as input, and generates estimates of nodes and edges. The node output $\epsilon_{\theta,\mathcal{X}}(\mathcal{G}_t, \bar{\boldsymbol{\Theta}}_t, t)$ corresponds to the gradient estimate of the node feature, capturing the direction of change of the node feature. The edge output $\epsilon_{\theta,\boldsymbol{\Theta}}(\mathcal{G}_t, \bar{\boldsymbol{\Theta}}_t, t)$ corresponds to the gradient estimate of the edge feature, capturing the direction of change of the edge feature. The training objective is optimized by minimizing the following loss function:

$$\begin{aligned} &\min_\theta \mathbb{E}_t\{w(t)\mathbb{E}_{\mathcal{G}_0}\mathbb{E}_{\mathcal{G}_t|\mathcal{G}_0}[\| \epsilon_{\theta,\mathcal{X}}(\mathcal{G}_t, \bar{\boldsymbol{\Theta}}_t, t) - \boldsymbol{\epsilon}_{\mathcal{X}} \|_2^2 \\ &+ \| \epsilon_{\theta,\boldsymbol{A}}(\mathcal{G}_t, \bar{\boldsymbol{\Theta}}_t, t) - \boldsymbol{\epsilon}_{\boldsymbol{\Theta}} \|_2^2]\} \end{aligned} \tag{38}$$

where $\boldsymbol{\epsilon}_{\mathcal{X}}$, $\boldsymbol{\epsilon}_\theta$ denote the Gaussian noise, $\mathcal{G}_t = (\alpha_t\tilde{\boldsymbol{\mathcal{X}}}_0 + \sigma_t\boldsymbol{\epsilon}_{\mathcal{X}}, \alpha_t\boldsymbol{\Theta}_0 + \sigma_t\boldsymbol{\epsilon}_{\boldsymbol{\Theta}})$.

To generate a parameterized SDE graph, a numerical solver is needed to simulate the SDE trajectory. Compared with the EM solver, the advantage of using a probability flow ODE solver is that it can utilize advanced ODE solving algorithms (e.g., the Runge–Kutta method [157]) to significantly reduce the number of sampling steps while maintaining high accuracy, thereby achieving fast sampling. In the specific implementation process, CDGS can apply a black-box ODE solver to the above probability flow ODE equations and generate high-fidelity graphs by optimizing the solution process. Therefore, the parameterized ODE of the graph is formally defined as follows:

$$\mathrm{d}\mathcal{G}_t/\mathrm{d}t = f(t)\mathcal{G}_t + \frac{g^2(t)}{2\sigma_t}\epsilon_\theta(\mathcal{G}_t, \bar{\boldsymbol{\Theta}}_t, t) \tag{39}$$

Traditional black-box ODE solvers [32] are usually unable to recognize and exploit the semilinear structure of probability flow ODEs, which results in many unnecessary calculations during the solution process, increasing the time complexity and computational cost. Therefore, CDGS further extends the fast solver based on the semilinear ODE structure. By redefining the time variable and parameterizing the graph diffusion process, the ODE solver is able to exploit the semilinear structure to optimize the solution process. Specifically, by introducing the time variable λ to change the time variable from t to λ, and $\lambda_t := \log(\alpha_t/\sigma_t)$, $t = t_\lambda(\lambda(t))$, $\hat{\mathcal{G}}_\lambda := \mathcal{G}_{t_\lambda(\lambda)}$, $\hat{\epsilon}_\theta(\hat{\mathcal{G}}_\lambda, \bar{\boldsymbol{\Theta}}'_\lambda, \lambda) := \epsilon_\theta(\mathcal{G}_{t_\lambda(\lambda)}, \bar{\boldsymbol{\Theta}}_{t_\lambda(\lambda)}, \lambda)$, CDGS is able to operate on a new time scale, making the equation form more suitable for fast solution using the semilinear structure. Therefore, the exact solution to the semilinear probability flow ODE is as follows:

$$\mathcal{G}_t = \frac{\alpha_t}{\alpha_s}\mathcal{G}_s - \alpha_t\int_{\lambda_s}^{\lambda_t} e^{-\lambda}\hat{\epsilon}_\theta(\hat{\mathcal{G}}_\lambda, \bar{\boldsymbol{\Theta}}'_\lambda, \lambda)\mathrm{d}\lambda \tag{40}$$

The initial graph $\tilde{\mathcal{G}}_{t_0} := \tilde{\mathcal{G}}_T = (\boldsymbol{\mathcal{X}}_T, \boldsymbol{\Theta}_T)$ is sampled from the prior distribution, with node features X_T and edge information A_T. At each time step t_i, the first-order GDPMS is used to iteratively calculate $\{\tilde{\mathcal{G}}_{t_i} = (\tilde{\mathcal{X}}_{t_i}, \bar{\Theta}_{t_i})\}_{i=1}^M$ as follows:

$$\begin{cases} \tilde{\boldsymbol{\mathcal{X}}}_{t_i} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{\mathcal{X}}}_{t_{i-1}} - \gamma_i\hat{\epsilon}_{\theta,\mathcal{X}}(\tilde{\mathcal{G}}_{t_{i-1}}, \bar{\boldsymbol{\Theta}}'_{t_{i-1}}, t_{i-1}) \\ \tilde{\boldsymbol{\Theta}}_{t_i} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{\boldsymbol{\Theta}}_{t_{i-1}} - \gamma_i\hat{\epsilon}_{\theta,\boldsymbol{\Theta}}(\tilde{\mathcal{G}}_{t_{i-1}}, \bar{\boldsymbol{\Theta}}'_{t_{i-1}}, t_{i-1}) \end{cases} \tag{41}$$

where $\gamma_i = \sigma_{t_i}(e^{\lambda_{t_i} - \lambda_{t_{i-1}}} - 1)$.

The probability flow ODE maps the input graph data into a latent space [20]. This latent space is determined by parameterized ODEs, and its properties allow flexible manipulation and transformation of the data [1]. The graph DPM solver, combined with gradient-guided techniques, enables efficient search and optimization in the latent space. The gradient-guided assisted solver uses the gradient information of the latent representation to help find the optimal solution that meets specific constraints. By defining similarity constraints in the latent space, CDGS can ensure that the molecules in the optimization process still retain their original chemical properties and functions. Combining the above methods, CDGS proposes an optimization process that can effectively find the optimal molecules that meet the similarity constraints in the latent space.

Specifically, CDGS trains a time-dependent graph property predictor $\boldsymbol{R}_\psi(\mathcal{G}_t, t)$ on the noise graph to estimate the properties of the graph at different time steps. The parameterized ODE solver in Eq. (39) is used to map the initial molecular graph to the encoding $\mathcal{G}_{t_\xi}$ in the latent space. This mapping process involves the adjustment of the time step t_ξ. According to the common latent space optimization method [40,55], the graph property predictor is used to predict the properties, and the properties of the graph are generated by gradient ascent optimization to generate the latent graph sequence $\{\mathcal{G}_{t_\xi}^k\}_{k=0}^K$. In the decoding process, a gradient-guided ODE is introduced to decode the latent graph to the molecular graph space. In this process, the gradient term of $\nabla_{\mathcal{G}}\boldsymbol{R}_{\boldsymbol{\psi}}(\mathcal{G}_t, t)$ is used to guide the sampling process to ensure that the generated graph has the desired high properties. The guided ODE can be modified from Eq. (39) as follows:

$$\begin{cases} \mathrm{d}\boldsymbol{\mathcal{X}}_t/\mathrm{d}t = f(t)\boldsymbol{\mathcal{X}}_t + \frac{g^2(t)}{2\sigma_t}[\epsilon_{\theta,\mathcal{X}} - r\sigma_t\nabla^*_{\boldsymbol{\mathcal{X}}}\boldsymbol{R}_{\boldsymbol{\psi}}] \\ \mathrm{d}\boldsymbol{\Theta}_t/\mathrm{d}t = f(t)\boldsymbol{A}_t + \frac{g^2(t)}{2\sigma_t}[\epsilon_{\theta,\boldsymbol{\Theta}} - r\sigma_t\nabla^*_{\boldsymbol{\Theta}}\boldsymbol{R}_{\boldsymbol{\psi}}] \end{cases} \tag{42}$$

Table 2
Complexity and scalability comparison of representative graph generative models. Here n is the number of nodes, m is the number of edges, d is the embedding dimension, and T is the number of diffusion steps.

Model Family	Time Complexity	Space Complexity	Scalability (w.r.t graph size)
VAE-based	$O(nd^2 + md)$	$O(nd + m)$	Moderate (scales to 10^5 nodes with sparse ops)
GAN-based	$O(nd^2 + md)$	$O(nd + m)$	Limited (training instability grows with graph size)
Autoregressive (AR)	$O(n^2 d)$	$O(n^2)$	Poor (sequential decoding hinders large graphs)
Diffusion-based	$O(T \cdot f(n, m, d))$, typically $O(Tn^2)$	$O(nd + m)$	Limited (large T and n^2 cost); improved by sparse/active node methods

4. Complexity and scalability analysis

In this section, we analyze the representative families of graph generative models, including VAE-based, GAN-based, autoregressive, and diffusion-based in terms of time complexity, space complexity, scalability with graph size, and theoretical convergence guarantees. Table 2 summarizes the complexity characteristics and convergence properties of these model families.

Time and space complexity. For VAE-based and GAN-based models, the dominant cost arises from message passing in graph neural network (GNN) encoders and decoders, yielding a time complexity of $O(nd^2 + md)$, where n is the number of nodes, m the number of edges, and d the embedding dimension. Their memory footprint is $O(nd + m)$ under sparse adjacency representation. Autoregressive models, however, require sequential prediction of node or edge probabilities, leading to $O(n^2 d)$ complexity and $O(n^2)$ space when modeling adjacency explicitly, which severely limits their scalability to large graphs. Diffusion-based approaches incur an iterative cost of $O(T \cdot f(n, m, d))$, typically $O(Tn^2)$, where T is the number of diffusion steps. This iterative nature makes diffusion models computationally demanding, though recent improvements such as EDGE [24] reduce the per-step cost to $O(\min(K^2, M))$ with K denoting active nodes and M the number of edges.

Scalability across graph sizes. VAE-based models generally scale to graphs of size up to 10^5 nodes when combined with sparse operations, whereas GAN-based methods are more restricted due to training instability. Autoregressive models scale poorly beyond small graphs because of their sequential decoding nature. Diffusion-based models can in principle leverage parallelism, but their dependence on large T and quadratic complexity in n poses challenges for million-scale networks. Sparse or subgraph-based variants represent promising solutions.

Convergence guarantees. VAE models benefit from the evidence lower bound (ELBO), ensuring a principled optimization target. Autoregressive models provide exact likelihood training but may suffer from exposure bias during inference. GAN-based models lack formal convergence guarantees due to adversarial dynamics. In contrast, diffusion models offer convergence under the framework of score matching, and continuous-time SDE/ODE variants provide stronger theoretical backing, though practical convergence often depends on numerical stability.

5. Implementation challenges

Despite the rapid progress of graph generative models, practical implementation still faces a number of challenges. First, large-scale training often leads to substantial memory consumption, especially for diffusion-based approaches that require iterative denoising and intermediate feature storage. Techniques such as gradient checkpointing, sparse tensor operations, and mini-batch subgraph sampling have been widely adopted to alleviate this issue. Second, autoregressive models typically suffer from slow sequential decoding, which hampers their scalability to graphs with tens of thousands of nodes. While parallelization strategies and heuristic node-ordering schemes have been proposed, their effectiveness remains dataset-dependent. Third, adversarial frameworks like GAN-based models are often unstable during training due to the delicate balance between the generator and discriminator, requiring careful tuning and multiple restarts in practice. Finally, diffusion-based methods, although flexible, incur high computational cost because of the large number of denoising steps; efficient variants such as fast-sampling schedulers or sparse/active-node diffusion have emerged as potential remedies.

6. Hyperparameter sensitivity

Another important practical aspect concerns the sensitivity of graph generative models to hyperparameter choices. Different families of models exhibit distinct critical parameters that directly impact both convergence and generation quality. For VAE-based methods, latent dimensionality and regularization strength significantly influence the trade-off between reconstruction fidelity and generalization. GAN-based methods are highly sensitive to the learning rate and the relative update frequency between generator and discriminator, which can easily lead to mode collapse if not carefully balanced. Autoregressive models depend heavily on node-ordering schemes and context window size, both of which affect sampling efficiency and likelihood estimation. Diffusion-based models are especially sensitive to the number of denoising steps T and the choice of noise schedule; insufficient steps may degrade sample quality, while excessive steps lead to prohibitively high computation costs. In practice, hyperparameter tuning often requires extensive empirical search, and small deviations may result in noticeable performance fluctuations.

7. Popular benchmark datasets

In graph generation research, it is very important to use standard datasets for algorithm verification and performance evaluation. As shown in Table 3, we have counted some commonly used graph generation datasets. Next, we briefly describe each graph generation dataset.

Community-small: Community-small [107,158] is a small dataset for studying and testing graph algorithms, especially in tasks such as community detection and graph clustering. The Community-small dataset contains obvious community structures. Each community represents a subgroup in the graph, which contains not only node and edge information but also node features (e.g., labels, attributes) and edge weights (e.g., connection strength).

Ego-small: Ego-small [107,159] is a small-scale dataset[2] that contains subgraphs from social networks that are centered on the ego and include its directly connected neighbor nodes. The Ego-small dataset contains detailed information between nodes (individuals) and their edges (relationships), including properties (e.g., node features and edge weights).

Grid: The Grid dataset [34] consists of multiple grid graphs whose nodes are arranged in a regular two-dimensional grid. Each node is usually connected to its four neighboring nodes, forming a regular grid structure. The Grid dataset contains grid graphs of different sizes, ranging from small 5x5 grids to large 100×100 grids.

QM9: QM9 [160] is a dataset widely used in quantum chemistry and molecular machine learning research, containing about 134,000 stable small molecules. These molecules are composed of five elements:

[2] https://github.com/ermongroup/GraphScoreMatching

Table 3
Popular benchmark dataset in graph generation. We counted the types of different data sets, etc.

Dataset	Dimensionality	Category	No. of Graphs (G)	No. of Nodes (N)
Community-small [107,158]	2D	Social	100	$11 < N < 20$
Ego-small [107,159]	2D	Social	200	$3 < N < 18$
Grid [34]	2D	Grid	100	$N <= 400$
QM9 [160]	3D	Bioinformatics/Molecular	130,831	$3 < N < 29$
ZINC250K [161]	3D	Bioinformatics/Molecular	249,456	$6 < N < 38$
Enzymes [162]	3D	Bioinformatics/Protein	600	$9 < N < 125$
SBM-27 [163]	2D	Social	200	$24 < N < 27$
Planar-60 [163,164]	2D	Social	200	$N = 60$
AIDS [165]	2D	Bioinformatics/Molecular	2000	–
Synthie [166]	2D	Social	300	$N = 100$
Proteins [167]	3D	Bioinformatics/Protein	1113	$N = 39.1$

hydrogen (H), carbon (C), oxygen (O), nitrogen (N), and fluorine (F). The QM9 dataset contains physical and chemical properties, such as molecular geometry (atomic coordinates), energy, Hall effect, electric dipole moment, vibration frequency, thermodynamic properties (such as enthalpy, and free energy), etc. The property data in the QM9 dataset are calculated by density functional theory (DFT), specifically using the B3LYP function and the 6-31G (2df, p) basis set.

ZINC250K: ZINC250K [161] is a dataset widely used in molecular machine learning and drug design research. It contains 250,000 small molecules screened from the ZINC database, which have diverse structural and chemical properties. The ZINC database is a free and publicly available database of chemical substances designed specifically for virtual screening and computer-aided drug design. The ZIN-C250K dataset contains a wide variety of compounds (e.g. different ring systems, functional groups, and stereochemistry).

Enzymes: The Enzymes dataset [162] consists of 600 graphs representing enzyme molecules. The nodes of each graph represent amino acid residues, and the edges represent the interactions between these residues. The Enzymes dataset contains 6 different types of enzymes, and each graph is classified according to the function of the enzyme. In addition to the graph structure, Enzymes also contain attribute information (e.g., the attributes of the nodes may include chemical properties, geometric information, etc.).

SBM-27: Stochastic Block Model - 27 (SBM-27) [163] is a dataset for studying community detection and graph clustering algorithms. SBM-27 is generated based on the stochastic block model, which is a statistical model commonly used to generate random graphs with community structure. The SBM-27 dataset contains 27 communities, each of which represents a subpopulation in the graph.

Planar-60: The Planar-60 dataset [163,164] consists of 60 planar graphs. A planar graph can be drawn on a plane without crossing edges. The graphs in the Planar-60 dataset have diverse structural features, including different numbers of nodes and edges, different connection patterns, etc.

AIDS: The AIDS dataset [165] is a well-known dataset in cheminformatics and is often used for the analysis and research of molecular graphs. The AIDS dataset (AIDS Antiviral Screen Data) was originally provided by the National Cancer Institute (NCI) of the United States to evaluate the inhibitory effect of compounds on HIV (human immunodeficiency virus) replication. The AIDS dataset contains the molecular structures and biological activity information of thousands of compounds. The biological activity data of each compound mainly describes its inhibitory effect on the HIV virus. The data is usually represented by binary classification (active or inactive) or multi-classification (according to the different degrees of inhibition).

Synthie: The Synthie dataset [166] is generated synthetically, and the structure of the graph, and the properties of nodes and edges can be adjusted as needed. The Synthie dataset contains multiple graphs, each of which represents a data sample. Since the generation method of the graph is controllable, researchers can create graph data of different difficulty and complexity by modifying the generation parameters to test.

Proteins: The Proteins dataset [167] consists of a set of protein molecules, each of which is represented as a graph in which nodes represent amino acids and edges represent interactions between these amino acids. The graphs in the Proteins dataset directly reflect the actual biological structure of protein molecules and have high biological relevance.

8. Evaluation metrics

8.1. Maximum mean discrepancy (MMD)

Evaluating the quality of generated graphs requires principled statistical distances that go beyond visual inspection or simple average statistics. Maximum Mean Discrepancy (MMD) has become one of the most widely used metrics for graph generation, as it compares two empirical distributions across all moments in a reproducing kernel Hilbert space (RKHS). Formally, with kernel function $k(\cdot,\cdot)$, the squared MMD between two distributions p and q is:

$$\begin{aligned}\mathrm{MMD}^2(p \parallel q) = \mathbb{E}_{x,x'\sim p}[k(x,x')] + \mathbb{E}_{y,y'\sim q}[k(y,y')] \\ - 2\mathbb{E}_{x\sim p,y\sim q}[k(x,y)]\end{aligned} \tag{43}$$

Directly computing distances over full graph distributions is intractable. Therefore, MMD is typically estimated on selected graph statistics $\mathcal{M} = \{M_1, \ldots, M_k\}$, including the degree distribution, clustering coefficient distribution, and motif or orbit counts. These statistics are embedded using kernels such as the Wasserstein or RBF kernel, enabling comparison of structural patterns between generated and real graphs.

Domain relevance. MMD is particularly suitable for social networks and citation networks, where capturing structural fidelity is crucial. In these domains, preserving degree distributions reflects the presence of hubs and power-law connectivity, while clustering coefficients characterize community structures and small-world effects. Similarly, motif counts provide insights into collaboration cliques or citation cycles that are critical for realistic graph synthesis. Thus, MMD not only serves as a general-purpose divergence measure, but also directly links to domain-specific challenges in social and citation networks: ensuring that generated graphs reproduce the characteristic structural statistics that underpin network connectivity, community formation, and information flow.

8.2. Fréchet ChemNet distance (FCD)

While MMD is widely applied for evaluating structural fidelity in social and citation networks, the **Fréchet ChemNet Distance (FCD)** is specifically designed to assess the quality of molecular graph generation. FCD measures the distance between the distribution $p(\cdot)$ of generated molecules and the distribution $p_w(\cdot)$ of real molecules, both embedded into the latent representation of a pretrained molecular property prediction network (ChemNet). By extracting activations from the penultimate layer of ChemNet, each molecule is represented as a feature vector. Assuming these feature vectors follow a Gaussian

Table 4
We summarize existing graph generation methods according to our classification method.

Method	Generative method						Generation process		Permutation	Compositional
	VAE	*GAN*	*Flow*	*RNN*	*EBM*	*Diffusion*	*One-shot*	*Sequential*	invariance	generation
GraphVAE [136]	✓	–	–	–	–	–	✓	–	✗	–
DeepGMG [168]	–	–	–	✓	–	–	–	✓	✗	–
CGVAE [169]	✓	–	–	–	–	–	–	✓	✗	–
MolGAN [139]	–	✓	–	–	–	–	✓	–	–	–
RVAE [170]	✓	–	–	–	–	–	✓	–	✗	–
GCPN [171]	–	✓	–	–	–	–	–	✓	✗	–
JT-VAE [40]	✓	–	–	–	–	–	–	✓	✗	–
MolecularRNN [35]	–	–	–	✓	–	–	–	✓	✗	–
GraphNVP [54]	–	–	✓	–	–	–	✓	–	✗	–
TGCD [172]	✓	–	–	–	–	–	✓	–	✗	–
GRF [173]	–	–	✓	–	–	–	✓	–	✗	–
GraphAF [133]	–	–	✓	–	–	–	–	✓	✗	–
HierVAE [174]	✓	–	–	–	–	–	–	✓	✗	–
MoFlow [55]	–	–	✓	–	–	–	✓	–	✗	–
GraphCNF [134]	–	–	✓	–	–	–	✓	–	✓	–
GraphEBM [175]	–	–	–	–	✓	–	✓	–	✓	✓
GraphRNN [34]	–	–	–	✓	–	–	–	✓	✗	–
GraphDF [56]	–	–	✓	–	–	–	–	✓	✗	–
GNF [176]	–	–	✓	–	–	–	✓	–	✗	–
EDP-GNN [107]	–	–	–	–	–	✓	✓	–	✓	–
CCGG [177]	✓	–	–	–	–	–	–	✓	✗	–
GDSS [32]	–	–	–	–	–	✓	✓	–	✓	–
ConGen [41]	–	✓	–	–	–	–	✓	–	✗	–
Digress [68]	–	–	–	–	–	✓	✓	–	✓	–
GraphARM [178]	✓	–	–	–	–	–	–	✓	✗	–
EB-GFN [179]	–	–	✓	–	–	–	✓	–	✗	–
GPrinFlowNet [180]	–	–	✓	–	–	–	✓	–	✗	–
GRAN [181]	–	–	–	✓	–	–	–	✓	✗	–
PPGN-Score [2]	–	–	–	–	–	✓	✓	–	✓	–
SubspaceDiff [182]	–	–	–	–	–	✓	✓	–	✓	–
WSGM [142]	–	–	–	–	–	✓	✓	–	✓	–
SPECTRE [163]	–	✓	–	–	–	–	✓	–	✗	–
SGGM [113]	–	–	–	–	–	✓	✓	–	✓	–
GSDM [125]	–	–	–	–	–	✓	✓	–	✓	–

distribution, the mean and covariance of the generated set (m, C) and the real set (m_w, C_w) are computed, and the Fréchet distance is given by:

$$d^2((m, C), (m_w, C_w)) = \| m - m_w \|^2 + \mathrm{Tr}(C + C_w - 2(CC_w)^{1/2}) \tag{44}$$

Intuitively, FCD evaluates both the quality (how close the generated molecules are to real ones in the feature space) and the diversity (whether the covariance of generated molecules matches that of the training set). In practice, a sufficiently large sample size (e.g., ≥5000 molecules) ensures stable estimation of mean and covariance, making FCD a reliable measure of molecular distribution alignment.

Domain relevance. FCD is particularly suitable for molecular graph generation, as it directly reflects the ability of generative models to capture realistic chemical features. Unlike generic graph-level metrics, FCD leverages task-specific pretrained networks that encode chemical semantics, ensuring that generated molecules not only resemble real molecules statistically but also preserve crucial biochemical properties. For example, a low FCD score indicates that generated molecules share similar pharmacophoric patterns and substructural motifs with real compounds, which is essential for downstream drug discovery and materials design tasks. Thus, FCD provides a domain-aware evaluation that bridges statistical similarity and chemical validity in generative modeling.

9. Experimental performance

As shown in Table 4, we first summarize the existing 33 methods from four aspects: generative method, generation process, permutation and compositional. Next, we will analyze the performance of these methods on different graph generation datasets.

As shown in Table 5, the GPrinFlowNet [180] method performs well in graph generation tasks, achieving the best performance compared to the baseline methods. In particular, compared to GDSS [32], a leading graph generation method using diffusion, GPrinFlowNet's model achieves significantly lower MMD scores on the Enzymes and Synthie datasets, by 2.4 and 3.0 times, respectively. These results show that GPrinFlowNet is not only able to generate graphs that are highly similar to the true graph structure, but also performs well in preserving the statistical properties of the graph. Compared with diffusion models such as GDSS, GPrinFlowNet has higher accuracy and efficiency in capturing the structure and statistical properties of the graph.

On the generic graph generation, we present the experimental results in Table 6, which clearly demonstrate the superior performance of GSDM [125]. GSDM demonstrates significant advantages over the autoregressive and one-shot baselines, fully demonstrating its status as the SOTA graph diffusion model. Moreover, GSDM significantly outperforms the autoregressive and one-shot baselines on multiple key performance metrics. The performance improvement is not only reflected in the generated quality, but also in the generated diversity and accuracy. GSDM performs diffusion across the entire graph data space, and by leveraging spectral methods, it more effectively captures the complex relationships and features in the graph structure, and can generate high-quality graphs in a shorter time. The experimental results further demonstrate that GSDM is particularly outstanding both in handling complex graph structures and in generating realistic and useful graph data.

On the molecules generation, GSDM [125] achieves the highest scores on multiple metrics in Table 7. In particular, GSDM performs particularly well on the NSPDK and FCD metrics. These high scores indicate that GSDM is able to generate molecules whose data distribution in chemical space and graph space is close to that of real molecules, demonstrating its strong ability in molecular design. The experimental

Table 5
Generation results on the conditional graph generation datasets.

		AIDS Real. $\|V\| \leq 95, \|C\| = 2$				Enzymes Real. $\|V\| \leq 125, \|C\| = 6$				Synthie Synthetic. $\|V\| \leq 100, \|C\| = 4$			
		Deg.↓	Clus.↓	Orbit↓	Avg.↓	Deg.↓	Clus.	Orbit↓	Avg.↓	Deg.↓	Clus.↓	Orbit↓	Avg.↓
Autoreg.	GraphRNN [34]	0.241	0.143	0.034	0.139	0.086	0.294	0.307	0.229	0.247	0.285	0.419	0.317
	GraphAF [133]	0.197	0.093	0.026	0.105	0.058	0.174	0.156	0.129	0.137	0.176	0.302	0.205
	GraphDF [56]	0.184	0.085	0.031	0.101	0.062	0.196	0.204	0.154	1.681	1.265	0.258	1.068
One-shot	GraphVAE [136]	0.358	0.284	0.127	0.256	1.249	0.687	0.381	0.772	1.554	1.074	0.232	0.953
	GNF [176]	0.224	0.159	0.018	0.133	–	–	–	–	–	–	–	
	EDP-GNN [107]	0.127	0.082	0.024	0.077	0.067	0.241	0.225	0.177	0.148	0.185	0.347	0.226
	CCGG [177]	0.097	0.074	0.035	0.068	0.043	0.125	0.117	0.095	0.107	0.159	0.236	0.167
	GDSS [32]	0.062	0.049	0.022	0.044	0.038	0.158	0.132	0.109	0.114	0.126	0.269	0.169
	CondGen [41]	0.138	0.115	0.032	0.095	0.065	0.184	0.213	0.154	0.151	0.162	0.295	0.202
	DiGress [68]	0.06	0.048	0.021	0.043	0.033	0.146	0.085	0.088	0.109	0.097	0.158	0.121
	GraphARM [178]	0.057	0.052	0.016	0.041	0.031	0.095	0.061	0.062	0.128	0.074	0.127	0.109
	EB-GFN [179]	0.094	0.087	0.035	0.072	0.079	0.213	0.227	0.173	0.152	0.164	0.341	0.219
	GPrinFlowNet [180]	0.046	0.031	0.012	0.029	0.027	0.062	0.046	0.045	0.048	0.042	0.079	0.056

Table 6
Generation results on the conditional generic graph generation datasets of baseline methods.

		Community-small Synthetic, $12 \leq \|V\| \leq 20$				Enzymes Real, $10 \leq \|V\| \leq 125$				Grid Synthetic, $100 \leq \|V\| \leq 400$			
		Deg.↓	Clus.↓	Orbit↓	Avg.↓	Deg.↓	Clus.↓	Orbit↓	Avg.↓	Deg.↓	Clus.↓	Orbit↓	Avg.↓
Autoreg.	DeepGMG [168]	0.220	0.950	0.400	0.523	–	–	–	–	–	–	–	–
	GRAPHARM [178]	0.034	0.082	0.004	–	0.029	0.054	0.015	–	–	–	–	
	GRAN [181]	0.001	0.084	0.028	0.020	–	–	–	–	0.001	0.004	0.002	0.016
	GraphRNN [34]	0.080	0.120	0.040	0.080	0.017	0.043	0.021	0.043	0.011	0.0	0.001	0.012
	GraphAF [133]	0.18	0.20	0.02	0.133	1.669	1.283	0.266	1.073	–	–	–	–
	GraphDF [56]	0.06	0.12	0.03	0.070	1.503	1.061	0.202	0.922	–	–	–	–
One-shot	GraphVAE [136]	0.350	0.980	0.540	0.623	1.369	0.629	0.191	0.730	1.619	0.0	0.919	0.846
	PPGN-Score [2]	0.081	0.237	0.284	0.200	–	–	–	–	–	–	–	–
	GNF [176]	0.200	0.200	0.110	0.170	–	–	–	–	–	–	–	–
	DiGress [68]	0.009	0.104	0.051	0.037	0.004	0.083	0.002	–	–	–	–	–
	EDP-GNN [107]	0.053	0.144	0.026	0.074	0.023	0.268	0.082	0.124	0.455	0.238	0.328	0.340
	SubspaceDiff [182]	0.057	0.098	0.012	0.056	0.037	0.099	0.018	0.051	0.124	0.013	0.090	0.076
	WSGM [142]	0.039	0.084	0.009	0.044	0.034	0.097	0.013	0.048	0.083	0.006	0.065	0.051
	GPrinFlowNet [180]	0.021	0.068	0.021	0.037	0.021	0.088	0.009	0.039	0.056	0.042	0.015	0.038
	SPECTRE [163]	0.008	0.1067	0.046	0.025	0.136	0.195	0.125	–	–	–	–	–
	GDSS [32]	0.045	0.086	0.007	0.046	0.026	0.102	0.009	0.046	0.111	0.005	0.070	0.062
	SGGM [113]	0.041	0.079	0.010	0.043	0.030	0.073	0.013	0.039	0.114	0.0	0.065	0.060
	SGGM+SLD [113]	0.035	0.071	0.006	0.037	0.022	0.062	0.007	0.030	0.103	0.0	0.053	0.052
	GSDM [125]	0.011	0.015	0.001	0.009	0.013	0.088	0.01	0.037	0.002	0.0	0.0	0.0007

Table 7
Generation results of baseline methods on the QM9 and ZINC250k datasets.

	Method	QM9					ZINC250k				
		Validity (%)↑	Val. w/o corr. (%)↑	NSPDK↓	FCD↓	Time (s)↓	Validity (%)↑	Val. w/o corr. (%)↑	NSPDK↓	FCD↓	Time (s)↓
Autoreg.	GraphAF [133]	100	67	0.020	5.268	$2.28e^3$	100	68	0.044	16.289	$5.72e^3$
	GraphEBM [175]	100	8.22	0.030	6.143	35.33	100	5.29	0.212	35.471	53.72
	GRAPHARM [178]	100	90.25	0.002	1.22	$1.52e^1$	100	88.23	0.055	16.26	$1.328e^2$
	GraphAF+FC [133]	100	74.43	0.021	5.625	$2.32e^3$	100	68.47	0.044	16.023	$5.91e^3$
	GraphDF [56]	100	82.67	0.063	10.816	$5.08e^4$	100	89.03	0.176	34.202	$5.87e^4$
	GraphDF+FC [56]	100	93.88	0.064	10.928	$4.72e^4$	100	90.61	0.177	33.546	$5.79e^4$
One-shot	MoFlow [55]	100	91.36	0.017	4.467	4.58	100	63.11	0.046	20.931	$2.59e^1$
	EDP-GNN [107]	100	47.52	0.005	2.680	$4.13e^3$	100	82.97	0.049	16.737	$8.41e^3$
	SGGM [113]	100	95.91	0.006	2.745	$4.93e^1$	100	97.28	0.018	13.931	$1.01e^3$
	SGGM+SLD [113]	100	97.35	0.004	2.593	–	100	98.32	0.014	11.379	$1.12e^3$
	SPECTRE [163]	100	87.3	0.163	47.96	3.3	100	90.2	0.109	18.44	103.1
	GDSS [32]	100	95.72	0.003	2.900	$1.06e^2$	100	97.01	0.019	14.656	$2.11e^3$
	GDSS-EM [32]	100	95.72	0.003	2.900	$1.06e^2$	100	97.01	0.019	14.656	$2.11e^3$
	GDSS-VP-EM [32]	100	95.72	0.003	2.900	$1.06e^2$	100	97.01	0.019	14.656*	$2.11e^3$
	GSDM [125]	100	99.90	0.003	2.650	$1.80e^1$	100	92.70	**0.017**	12.956	$4.59e^1$

results demonstrate that the proposed GSDM is not only suitable for general graph generation but also for molecular design. One of the main advantages of GSDM is its efficiency in generating molecules compared to other diffusion models such as EDP-GNN [107] and GDSS [32]. Table 7 also shows the time (in seconds) taken to generate 10,000 molecules. The results show that GSDM takes significantly less time than EDP-GNN and GDSS during inference. For example, GSDM takes only 18.02 s and 45.91 s to generate 10,000 molecular graphs on the QM9 [160] and Zinc250k [161] datasets, while GDSS takes 1060 s and 2110 s, respectively. This means that GSDM is 58 times and 46 times faster than GDSS. In summary, the results fully demonstrate that GSDM performs well in molecular design and other graphics generation

Table 8
Experimental results on protein datasets.

	Method	Proteins									
		Deg. ↓	Clus. ↓	Orbit ↓	Spectral ↓	Wavelet ↓	Ratio ↓	Unique ↑	Novel ↑	Uniq. & Nov. ↑	t (s) ↓
	Training set	0.0003	0.0068	0.0032	0.0009	0.0003	1.0	100.0	—	—	—
Autoreg.	GraphRNN [34]	0.0040	0.1475	0.5851	0.0152	0.0530	82.3	100.0	100.0	100.0	36.41
	GRAN [181]	0.0479	0.1234	0.3458	0.0125	0.0341	82.7	100.0	100.0	100.0	11.68
One-shot	MolGAN [139]	0.0008	0.0644	0.0081	0.0021	0.0012	4.2	97.3	100.0	97.3	0.003
	GG-GAN (RS) [183]	0.4727	0.1772	0.7326	0.4102	0.6278	875.8	100.0	100.0	100.0	0.482
	GG-GAN [183]	0.5192	0.5220	0.7326	0.3996	0.6157	906.5	100.0	100.0	100.0	0.485
	SPECTRE [163]	0.0056	0.0843	0.0267	0.0052	0.0118	16.9	100.0	100.0	100.0	0.507
	SPECTRE (real spectra) [163]	0.0013	0.0469	0.0287	0.0020	0.0022	6.0	100.0	100.0‡	100.0‡	0.485

tasks, significantly outperforming other baseline methods. Through the spectral diffusion method, GSDM achieves higher generation quality and efficiency.

As shown in Table 8, most GAN [183] baseline methods face training difficulties on larger real-world protein graph datasets. Notably, MolGAN [139] excels in generating graphs with excellent statistical metrics. However, in-depth analysis reveals that graphs generated by MolGAN are often just slight variants of a few graphs. In fact, the average dissimilarity of edges between any two graphs generated by MolGAN is only 17.6%. In the SPECTRE [163] method, mode collapse, a common problem of GANs, is successfully avoided by applying a teacher forcing mechanism and adjusting the generator and discriminator according to real spectral features. Specifically, Teacher Forcing helps the generator better learn to generate samples of real data by introducing real data as a reference in training. Adjusting the generator and discriminator further enhances the model's generation ability by optimizing the model's adaptability to data features and avoiding the phenomenon of generating monotonous graphs.

10. Efficiency

Fig. 5 presents a comparative study of the sampling efficiency of several representative deep generative graph models. The evaluation is conducted by recording the average wall-clock time required for each model to generate a single graph, with results averaged over 128 independent runs for consistency. The figure reports sampling time as a function of graph size, measured both in terms of the average number of nodes and the average number of edges of the generated graphs. It can be observed that most neural baselines, such as GDSS, DiscDDPM, and DiGress, are only able to generate graphs for relatively small datasets like Community (average 110 nodes) and Ego (average 144 nodes). GraphRNN, in contrast, can scale to larger graphs but incurs considerably higher sampling costs, reflecting the sequential nature of its autoregressive decoding. GraphCNF demonstrates strong efficiency on small datasets, with sampling times that are among the lowest, while diffusion-based methods show relatively slower performance due to their iterative denoising process. A noteworthy observation from the results is that the sampling time does not always increase monotonically with the number of nodes. For example, Ego graphs, despite having more nodes on average than Community graphs, require less generation time under certain models. This is attributed to the fact that the computational burden of some methods scales more directly with the number of edges, and Ego graphs are typically much sparser than Community graphs.

Overall, the comparative results highlight that while autoregressive methods are severely limited by scalability, flow-based approaches such as GraphCNF achieve strong efficiency on smaller graphs. Diffusion-based models offer greater modeling flexibility but face higher sampling costs. These findings underscore the trade-offs between scalability and sampling speed across different categories of generative graph models.

11. Applications

In this section, we will delve into real-world applications of graph generation techniques. We will focus on five specific application examples: molecular design, recommender systems, protein design, community generation, and program synthesis.

11.1. Molecular design

Molecular design is of great significance in application fields such as drug development, material science, environmental protection, agriculture, and the food industry. Through molecular design, new molecules and new materials with specific functions and excellent properties can be developed. In the task of molecular generation, generative design usually has two main goals: (1) graph generation methods should generate syntactically valid molecules; (2) the generated mole-cules should have certain specific properties. To ensure the validity of generated molecules, sequential generation strategies can be implemented by adding valence checks in each intermediate generation step. In existing work, EDM [4] introduces an E(n)-equivariant diffusion model tailored for generating 3D molecular structures, which handles both continuous (e.g., atomic coordinates) and categorical data by integrating equivariant properties into the denoising process to ensure that the generated molecular conformations respect the inherent symmetries of 3D space (e.g., rotations and translations). GeoDiff [5] proposes a geometric diffusion method for generating molecular conformations that focuses on modeling the Boltzmann distribution of molecular structures, ensuring that the generated conformations are physically plausible and energetically stable. GeoDiff exploits the SE(3) group to preserve rotational and translational invariance during diffusion and generation. As shown in Fig. 6, GeoDiff uses a graph diffusion model to generate highly realistic molecular conformations that have typical characteristics of drug-like molecules. We can clearly see multiple molecular conformation examples generated by GeoDiff, each of which shows a different molecular arrangement and structure. The visualization results not only verify the effectiveness of the GeoDiff model, but also demonstrate its potential application value in drug discovery and materials science. DiffLinker [6] generates molecular linkers conditioned on 3D fragments using a 3D equivariant diffusion model that incorporates symmetry considerations, allowing it to efficiently generate linkers that are consistent with the spatial arrangement of the input molecular fragments.

11.2. Recommender systems

Traditional recommendation systems may not fully capture the dynamic changes in user preferences, while diffusion models can provide more accurate recommendation results by generating and predicting users' potential interests through diverse data. In existing work, CF-Diff [184] proposes a new collaborative filtering method based on the diffusion model, which uses high-order connectivity information to improve the recommendation accuracy. SDRM [185] introduces a score-based diffusion recommendation module to generate synthetic

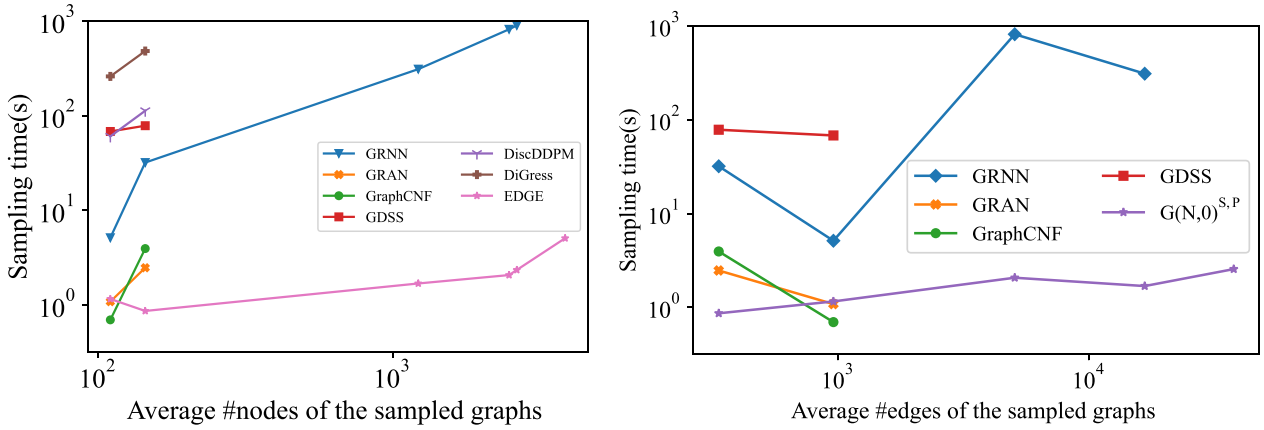

Fig. 5. Sampling speed comparison over different models.

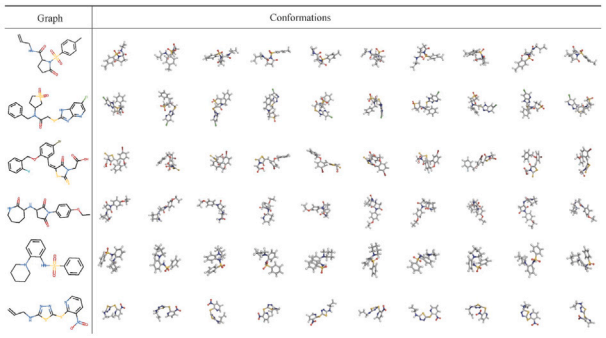

Fig. 6. Visualization of drug-like conformations generated by GEODIFF demonstrates the model's capability in generating molecular structures [5].

Fig. 7. DiffRec shows examples of user watch lists and recommended movies. In these examples, the real movie entries are marked with red boxes to facilitate comparison with the movie lists generated by the recommendation system [187].

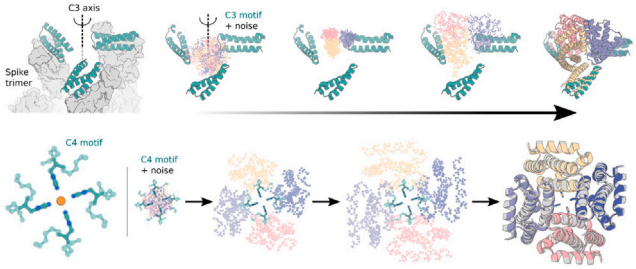

Fig. 8. The RFdiffusion track constructed a symmetric oligomer through multiple iterations. This process demonstrates the ability of RFdiffusion to gradually generate and optimize molecular structures to gradually meet the design goals [188].

data for privacy-preserving recommendations. SDRM captures complex patterns in real-world datasets, enabling accurate recommendations while protecting user privacy. G-Diff [186] emphasizes that diffusion models can better capture the uncertainty and potential preferences in user behavior compared to traditional generative models such as VAE [42,47,136] and GAN [41,61]. G-Diff learns the probability distribution of user behavior by adding noise to the target items, thereby achieving better sequence recommendations. As shown in Fig. 7, for the sake of brevity and clarity, G-Diff only shows the top five recommendation lists generated by three different random seeds. These recommendation lists show the movies recommended by the system under different initial conditions, which helps to understand the performance and variability of the recommendation algorithm in practical applications, and also helps to evaluate the accuracy and consistency of the recommendation system in meeting user viewing preferences.

11.3. Protein design

Designing proteins with specific functions through graph generation technology can promote the development of biotechnologies (e.g., enzyme engineering, and antibody design), and achieve efficient biocatalysis and disease treatment. In existing work, GDD [7] discusses how to use guided diffusion models to design proteins by sampling from non-norma-lized density functions. GDD emphasizes the efficiency of guided diffusion models in generating low-energy protein samples via fixed-length Markov chains. RFdiffusion [188] combines diffusion models with structure prediction networks to improve the accuracy of protein design. This approach uses diffusion models to create biologically plausible proteins, demonstrating their potential to generating diverse and structurally plausible protein sequences. EvoDiff [189] combines evolutionary-scale data with the regulatory power of diffusion models to generate high-fidelity, diverse proteins with broad functionality and structural plausibility, pushing the boundaries of traditional structure-based protein design methods. As shown in Fig. 8, RFdiffusion successfully designed oligomers that met the C4 symmetry and Ni2+ binding requirements, verifying the effectiveness of RFdiffusion in designing and generating molecular structures that meet theoretical expectations.

11.4. Community generation

Graph generation helps simulate complex community structures and dynamic changes in community generation, so as to better understand and analyze the behavior and characteristics of actual community networks. By generating representative graphs, we can fill in the missing parts in real data and enhance the quality and diversity of data sets.

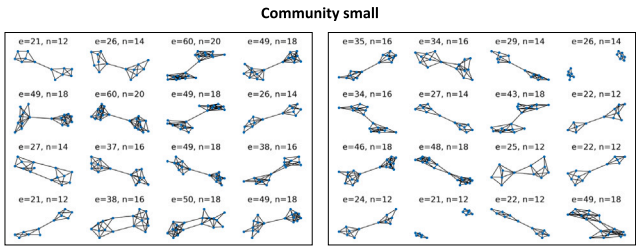

Fig. 9. GDSS shows the network structure of a small dataset of real communities. The graph includes nodes and their edge connections within the community, as well as the boundaries between different communities [32].

This is especially important for community detection and analysis, because real community data may be incomplete or noisy. Graph generation models can generate potential community structures and predict how communities evolve over time, which has practical application value for social network analysis. As shown in Fig. 9, GDSS [32] is a technique for graph data synthesis and sampling, and the graphs it generates are intended to simulate and predict possible community structures. The generated graphs usually reflect the community organization and node connectivity produced by the model under given parameters. Compared with the graphs of real datasets, the graphs generated by GDSS may show more experimental structural features.

11.5. Program synthesis

Different from traditional generation models, diffusion models can continuously verify and adjust the code during the generation process to ensure that the final generated code meets syntactic and semantic correctness. DST [190] introduces a neural diffusion model that operates on syntax trees to iteratively improve code while maintaining syntactic validity. DST addresses the limitations of autoregressive models by allowing iterative editing based on runtime feedback. As shown in Fig. 10, the leftmost column shows real renderers from the test set. The following columns show renderers generated by different methods, including DST and other baseline methods. Each column shows the rendering effect of a method in the corresponding graphics language. By comparing these renderers, we can see the differences in the accuracy and detail of the generated graphics of each method. The renderer of the DST method shows its advantages in graphics generation, especially in the precise adjustment of details and the matching degree with the real program. DST can generate more sophisticated graphics, better fit the real renderer in the test set, and show higher accuracy and realism.

12. Practical considerations across domains

When applying graph generative models to different domains, several domain-specific considerations arise. In molecular and materials science, ensuring chemical validity is essential, which often requires the incorporation of hard constraints or rule-based post-processing modules within generative pipelines. In contrast, applications in social networks and citation graphs prioritize scalability, where sparsity-aware message passing and subgraph sampling strategies become critical to handle million-scale graphs. Temporal or dynamic graphs, such as transaction networks, demand architectures that explicitly model time-evolving edges; ODE- and SDE-based formulations have been adapted to capture such dynamics. Moreover, the evaluation protocols also differ: molecular generation relies heavily on property-based metrics (e.g., QED, logP), while community network generation emphasizes structural similarity (e.g., degree distribution, clustering coefficient). These practical differences suggest that a "one-size-fits-all" solution remains elusive, and effective deployment requires careful alignment between model design, hyperparameter settings, and domain constraints.

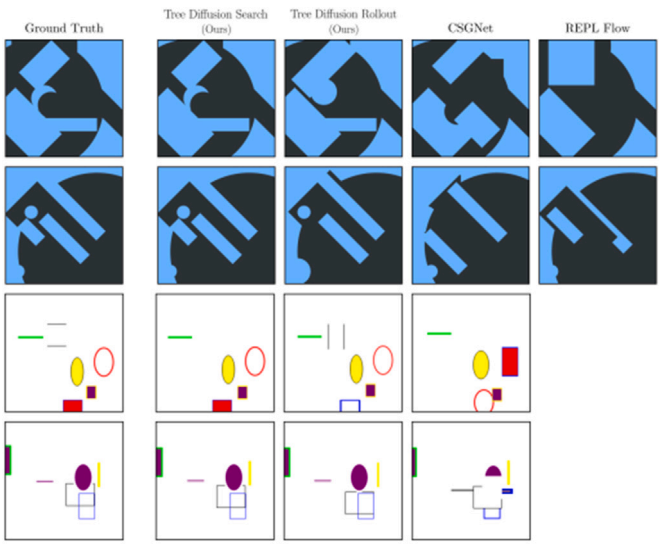

Fig. 10. Qualitative examples showing DST and baseline methods on two reverse graphics languages, CSG2D (top two rows) and TinySVG (bottom two rows) [190].

13. Future directions

Training Objectives and Optimization Problems. Although the Evidence Lower Bound (ELBO) has been widely adopted in graph generation tasks, its theoretical alignment with the true negative log-likelihood remains insufficiently understood. Optimizing ELBO serves only as a proxy for variational inference and does not necessarily guarantee convergence to the optimal generative distribution, particularly in high-dimensional structured data like graphs. To address this gap, several concrete research directions can be pursued: **(1) Theoretical calibration of ELBO.** A systematic approach is to derive tighter bounds by augmenting the standard ELBO with correction terms that explicitly account for graph-specific structures. For instance, one could integrate Rényi divergence or χ^2-divergence based bounds into the training objective, and empirically evaluate whether these bounds yield better alignment with true likelihoods on benchmark datasets. **(2) Hierarchical and domain-aware objectives.** Current objectives often treat nodes and edges uniformly, ignoring hierarchical or domain-specific dependencies. A practical pipeline would be: (i) decompose graphs into hierarchical units (e.g., communities in social graphs, functional groups in molecular graphs), (ii) define local reconstruction losses within each unit, and (iii) aggregate them into a global objective weighted by structural importance. This ensures that training respects both local fidelity and global coherence. **(3) Adaptive noise-signal balancing.** During diffusion, the challenge is to model noise while preserving meaningful signal. A concrete strategy is to adopt a curriculum-based schedule: start with high noise levels to encourage exploration, then gradually reduce noise according to topology-aware criteria (e.g., node degree distribution, motif frequency) so that crucial structural information is preserved. This can be implemented by dynamically adjusting the schedule conditioned on graph statistics. **(4) Hybrid loss design.** Recent advances in score matching and denoising diffusion suggest combining multiple objectives. An effective workflow could be: (i) reconstruction error enforces fidelity to input structure, (ii) adversarial regularization encourages realistic sample distribution, and (iii) structural constraints (e.g., energy-based priors in molecular graphs, modularity in social networks) guide the model toward domain validity. Such hybrid objectives can be optimized jointly, with adaptive weighting to prevent dominance of any single component. **(5) Evaluation and iterative refinement.** To validate these objectives, one can adopt a train–evaluate–refine loop: train with the proposed hybrid loss,

evaluate with both likelihood-based metrics (ELBO gap, NLL approximation) and domain metrics (MMD for social/citation networks, FCD for molecular graphs), and iteratively refine the objective weights or noise schedules until both statistical and domain-specific performance criteria are satisfied.

Challenges of 2D to 3D Graph Generation. As graph generation research evolves from 2D to 3D domains, new challenges arise in modeling spatially complex structures and ensuring geometric fidelity. Unlike 2D graphs, 3D graphs involve intricate interdependencies between node coordinates, bond angles, and geometric constraints, which are critical in molecular conformation and protein folding. To systematically address these challenges, several concrete approaches can be considered: **(1) Domain-aware evaluation metrics.** Traditional structural metrics (degree distribution, clustering coefficient) are insufficient for 3D graphs. A practical evaluation pipeline should integrate: (i) RMSD to measure deviation from reference conformations, (ii) steric clash scores to penalize physically invalid overlaps, and (iii) energy-based metrics from molecular dynamics or force-field simulations (e.g., MMFF94, CHARMM) to assess thermodynamic plausibility. This ensures that evaluation reflects both structural fidelity and physical validity. **(2) Equivariance and invariance modeling.** Generative models must respect SE(3) symmetries so that generated graphs remain consistent under rotation and translation. A concrete modeling strategy is: (i) adopt SE(3)-equivariant GNNs or tensor field networks for message passing, (ii) ensure that coordinate updates preserve rotational/translation invariance, and (iii) combine with attention mechanisms to capture long-range dependencies in proteins or large molecules. **(3) Geometric constraint preservation.** A key step in 3D generation is maintaining bond lengths, bond angles, and torsion angles throughout the denoising trajectory. This can be achieved by: (i) embedding constraints into the loss function (e.g., harmonic penalties for bond lengths), (ii) projecting generated coordinates back onto physically valid manifolds after each diffusion step, and (iii) using constraint-satisfaction layers that correct invalid geometries before sampling the next state. **(4) Incorporating physical priors.** Differentiable physics simulators and energy-based models can serve as inductive priors. A feasible workflow is: (i) generate coarse 3D structures using diffusion or score-based models, (ii) refine them with energy minimization guided by force fields, and (iii) feed the refined structures back into the generative loop for iterative improvement. This "generate–refine–retrain" pipeline links deep generative learning with molecular simulation.

Impact of Changes in Data Distribution. Another promising direction lies in addressing the challenges posed by distribution shifts between training and deployment data. In real-world applications such as molecular design or social network modeling, the data encountered at test time often exhibit domain-specific structures and constraints not fully captured during training. To enhance generative fidelity, future work should explore integrating domain-specific priors into the diffusion process. For instance, in molecular generation tasks, incorporating chemical valence constraints, ring closure rules, or energy-based physical priors into the denoising model can ensure that generated molecules are not only structurally valid but also chemically feasible. Similarly, in social network or community graph generation, embedding structural priors such as modularity, community detection statistics, or degree distributions can improve realism. Moreover, addressing distributional shifts requires robust modeling strategies, such as out-of-distribution (OOD) detection during sampling, domain-adaptive training objectives, or Bayesian diffusion models that explicitly model uncertainty under varying data regimes. These strategies will not only improve the stability and generalizability of graph diffusion models but also provide stronger guarantees for their deployment in downstream applications with dynamic and heterogeneous data distributions.

Permutation Invariance and Graph Alignment. Permutation invariance has traditionally been regarded as a fundamental property in graph generative models, ensuring that the model outputs remain consistent under node reordering. Many prior works emphasize this property as crucial for learning stable representations and avoiding overfitting to arbitrary node indices. However, emerging studies suggest that strict permutation invariance may, in some cases, limit the expressive power and generative performance of models. Recent research has demonstrated that relaxing permutation invariance can actually benefit generation tasks, especially in cases where alignment between graph structures is meaningful or where canonical node orderings can be inferred from data. For instance, SwingNN [191] introduces a permutation-sensitive diffusion model and shows that incorporating alignment-aware architectures can improve sample quality in molecular generation. Similarly, Laabid et al. [192] highlight that in retrosynthesis, aligning the node ordering between reactants and products is essential for achieving chemically plausible transformations. Even in large-scale structural prediction tasks like AlphaFold 3 [193], canonicalization of node positions is used to facilitate accurate biomolecular interaction modeling. These insights reveal a paradigm shift: rather than enforcing strict permutation invariance, modern graph generative models may benefit from incorporating task-specific or domain-aware alignment strategies. This includes learning permutation-sensitive features, introducing alignment loss terms, or leveraging canonical graph orderings derived from heuristics or auxiliary models. Future work in graph diffusion should therefore carefully assess when permutation invariance is beneficial and when controlled relaxation or alignment can provide superior performance.

Scalability of GDMs. A critical limitation of current graph diffusion models (GDMs) is their poor scalability to large-scale graphs such as social or citation networks. Most diffusion-based approaches require repeated pairwise computations across all nodes during the denoising process, while autoregressive models generate edges sequentially. Both designs lead to quadratic or worse computational costs, which become infeasible when the number of nodes reaches millions. Several practical workarounds have been proposed in recent studies. One direction is sparse or active-region diffusion, where models restrict computation to a subset of nodes or candidate edges that are most likely to change, instead of processing the entire dense adjacency at each step. This reduces overhead substantially while maintaining accuracy on large, sparse graphs. Another line of work focuses on subgraph sampling: large graphs are divided into overlapping subgraphs for training, and later merged through consistency constraints across boundaries. Although this strategy scales linearly with the number of subgraphs and allows parallelization, it must be carefully designed to preserve global structural properties. A complementary solution is hierarchical generation, which builds graphs in multiple stages. Models first generate a coarse skeleton at the community level and then refine each community in detail. This approach not only improves efficiency but also naturally fits the modular structure often observed in social and biological networks. Meanwhile, diffusion-step reduction techniques, such as distillation or adaptive noise schedules, shorten the number of denoising iterations required, accelerating both training and inference. Other engineering-oriented strategies include using low-rank or block-sparse representations of adjacency matrices to cut down redundant computations, as well as distributed and parallel training frameworks to leverage multiple GPUs or nodes effectively. Finally, edge-budget control and negative sampling strategies have been explored to avoid scoring all possible node pairs, instead focusing on the most informative edges. While these methods collectively alleviate scalability concerns, each comes with trade-offs, for example, sparse methods risk missing rare but important edges, subgraph sampling may lose global consistency, and hierarchical generation relies heavily on the quality of community detection. Achieving true scalability without compromising fidelity and stability remains an open research challenge, but the combination of these strategies provides a promising path forward.

Out-of-Distribution (OOD) Robustness. Another critical limitation of current GDMs lies in their vulnerability to distribution shift when applied to real-world data. While benchmarks typically assume that training and test graphs come from the same distribution, real

scenarios differ: social networks evolve over time, molecular libraries expand with novel scaffolds, and citation graphs shift with emerging domains. Such out-of-distribution (OOD) settings often cause severe degradation in generation quality, raising concerns about the reliability of current evaluation practices. **Existing mitigation strategies.** Several approaches have been explored to address OOD robustness: (1) *Self-supervised and pretraining methods* leverage large heterogeneous graph corpora to learn generalizable representations that transfer better across domains [194]. (2) *Domain adaptation techniques* explicitly align source and target distributions, for example using adversarial objectives or feature normalization to reduce domain gaps [195]. (3) *Uncertainty-aware modeling* equips GDMs with Bayesian layers or ensemble scoring to quantify confidence and detect OOD samples [196].

14. Conclusion

As one of the most advanced generative methods, diffusion-based methods have made significant progress in graph generation tasks, especially in areas (e.g., molecular design and material synthesis). Therefore, we provide a comprehensive review of diffusion-based graph generation methods. We first briefly review some traditional graph generation methods. Second, we analyze the different paradigms of diffusion methods, including how to apply diffusion models to graph structures. Third, we elaborate on the application of diffusion-based graph generation methods in various tasks, including their performance on popular datasets. By comparing the effects of different methods in practical applications, we can evaluate their effectiveness and advantages in specific tasks. Fourth, we describe the specific performance of diffusion-based graph generation methods in practical applications. Finally, we look forward to future research directions and challenges. We discuss the limitations of current methods and possible future research directions, including how to solve existing problems and explore new application areas and technological innovations.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No. 62472165), the Hunan Provincial Natural Science Foundation General Project, China (Grant No. 2025JJ50380), and the Hunan Provincial Natural Science Foundation Youth Project, China (Grant No. 2025JJ60420).

Data availability

Data will be made available on request.

References

[1] A.Q. Nichol, P. Dhariwal, Improved denoising diffusion probabilistic models, in: International Conference on Machine Learning, PMLR, 2021, pp. 8162–8171.
[2] J. Austin, D.D. Johnson, J. Ho, D. Tarlow, R. Van Den Berg, Structured denoising diffusion models in discrete state-spaces, Adv. Neural Inf. Process. Syst. 34 (2021) 17981–17993.
[3] Y. Song, S. Ermon, Generative modeling by estimating gradients of the data distribution, Adv. Neural Inf. Process. Syst. 32 (2019).
[4] E. Hoogeboom, V.G. Satorras, C. Vignac, M. Welling, Equivariant diffusion for molecule generation in 3d, in: International Conference on Machine Learning, PMLR, 2022, pp. 8867–8887.
[5] M. Xu, L. Yu, Y. Song, C. Shi, S. Ermon, J. Tang, GeoDiff: A geometric diffusion model for molecular conformation generation, in: International Conference on Learning Representations, 2022.
[6] I. Igashov, H. Stärk, C. Vignac, A. Schneuing, V.G. Satorras, P. Frossard, M. Welling, M. Bronstein, B. Correia, Equivariant 3D-conditional diffusion model for molecular linker design, Nat. Mach. Intell. (2024) 1–11.
[7] N. Gruver, S. Stanton, N. Frey, T.G. Rudner, I. Hotzel, J. Lafrance-Vanasse, A. Rajpal, K. Cho, A.G. Wilson, Protein design with guided discrete diffusion, Adv. Neural Inf. Process. Syst. 36 (2024).
[8] H. Cao, C. Tan, Z. Gao, Y. Xu, G. Chen, P.-A. Heng, S.Z. Li, A survey on generative diffusion models, IEEE Trans. Knowl. Data Eng. (2024).
[9] L. Yang, Z. Huang, Z. Zhang, Z. Liu, S. Hong, W. Zhang, W. Yang, B. Cui, L. Zhang, Graphusion: Latent diffusion for graph generation, IEEE Trans. Knowl. Data Eng. (2024).
[10] L. Sun, Z. Zhang, J. Zhang, F. Wang, H. Peng, S. Su, P.S. Yu, Hyperbolic variational graph neural network for modeling dynamic graphs, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, (5) 2021, pp. 4375–4383.
[11] H. Cao, Z. Zhang, L. Sun, Z. Wang, Inductive and irregular dynamic network representation based on ordinary differential equations, Knowl.-Based Syst. 227 (2021) 107271.
[12] Y. Shou, T. Meng, W. Ai, N. Yin, K. Li, A comprehensive survey on multi-modal conversational emotion recognition with deep learning, 2023, arXiv preprint arXiv:2312.05735.
[13] Y. Shou, T. Meng, W. Ai, S. Yang, K. Li, Conversational emotion recognition studies based on graph convolutional neural networks and a dependent syntactic analysis, Neurocomputing 501 (2022) 629–639.
[14] Y. Shou, T. Meng, W. Ai, F. Zhang, N. Yin, K. Li, Adversarial alignment and graph fusion via information bottleneck for multimodal emotion recognition in conversations, Inf. Fusion 112 (2024) 102590.
[15] W. Ai, Y. Shou, T. Meng, K. Li, Der-gcn: Dialog and event relation-aware graph convolutional neural network for multimodal dialog emotion recognition, IEEE Trans. Neural Netw. Learn. Syst. 36 (3) (2024) 4908–4921.
[16] Y. Shou, H. Liu, X. Cao, D. Meng, B. Dong, A low-rank matching attention based cross-modal feature fusion method for conversational emotion recognition, IEEE Trans. Affect. Comput. (2024).
[17] Y. Shou, W. Ai, T. Meng, N. Yin, Graph information bottleneck for remote sensing segmentation, 2023, arXiv preprint arXiv:2312.02545.
[18] T. Meng, Y. Shou, W. Ai, J. Du, H. Liu, K. Li, A multi-message passing framework based on heterogeneous graphs in conversational emotion recognition, Neurocomputing 569 (2024) 127109.
[19] Y. Song, S. Ermon, Improved techniques for training score-based generative models, Adv. Neural Inf. Process. Syst. 33 (2020) 12438–12448.
[20] J. Ho, A. Jain, P. Abbeel, Denoising diffusion probabilistic models, Adv. Neural Inf. Process. Syst. 33 (2020) 6840–6851.
[21] J. Hu, B. Hooi, S. Qian, Q. Fang, C. Xu, MGDCF: Distance learning via Markov graph diffusion for neural collaborative filtering, IEEE Trans. Knowl. Data Eng. (2024).
[22] Y. Zhang, Z. Bao, Y. Li, B. Zheng, X. Wang, From a timeline contact graph to close contact tracing and infection diffusion intervention, IEEE Trans. Knowl. Data Eng. (2024) 1.
[23] P. Bao, R. Yan, C. Yang, Popularity prediction via modeling temporal dependencies on dynamic evolution process, IEEE Trans. Knowl. Data Eng. (2024).
[24] X. Chen, J. He, X. Han, L.-P. Liu, Efficient and degree-guided graph generation via discrete diffusion modeling, in: Proceedings of the 40th International Conference on Machine Learning, 2023, pp. 4585–4610.
[25] M. Zhang, M. Qamar, T. Kang, Y. Jung, C. Zhang, S.-H. Bae, C. Zhang, A survey on graph diffusion models: Generative ai in science for molecule, protein and material, 2023, arXiv preprint arXiv:2304.01565.
[26] H. Chen, C. Xu, L. Zheng, Q. Zhang, X. Lin, Diffusion-based graph generative methods, 2024, arXiv preprint arXiv:2401.15617.
[27] D. Ghio, Y. Dandi, F. Krzakala, L. Zdeborová, Sampling with flows, diffusion, and autoregressive neural networks from a spin-glass perspective, Proc. Natl. Acad. Sci. 121 (27) (2024) e2311810121.
[28] G. Papamakarios, E. Nalisnick, D.J. Rezende, S. Mohamed, B. Lakshminarayanan, Normalizing flows for probabilistic modeling and inference, J. Mach. Learn. Res. 22 (57) (2021) 1–64.
[29] Y. Chen, J. Liu, L. Peng, Y. Wu, Y. Xu, Z. Zhang, Auto-encoding variational bayes, Camb. Explor. Arts Sci. 2 (1) (2024).
[30] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, Commun. ACM 63 (11) (2020) 139–144.
[31] Y. Song, S. Garg, J. Shi, S. Ermon, Sliced score matching: A scalable approach to density and score estimation, in: Uncertainty in Artificial Intelligence, PMLR, 2020, pp. 574–584.
[32] J. Jo, S. Lee, S.J. Hwang, Score-based generative modeling of graphs via the system of stochastic differential equations, in: International Conference on Machine Learning, PMLR, 2022, pp. 10362–10383.
[33] Y. Shou, T. Meng, W. Ai, K. Li, Dynamic graph neural ODE network for multi-modal emotion recognition in conversation, in: Proceedings of the 31st International Conference on Computational Linguistics, 2025, pp. 256–268.

[34] J. You, R. Ying, X. Ren, W. Hamilton, J. Leskovec, Graphrnn: Generating realistic graphs with deep auto-regressive models, in: International Conference on Machine Learning, PMLR, 2018, pp. 5708–5717.

[35] M. Popova, M. Shvets, J. Oliva, O. Isayev, MolecularRNN: Generating realistic molecular graphs with optimized properties, 2019, arXiv preprint arXiv:1905.13372.

[36] D. Bacciu, A. Micheli, M. Podda, Edge-based sequential graph generation with recurrent neural networks, Neurocomputing 416 (2020) 177–189.

[37] N. Goyal, H.V. Jain, S. Ranu, Graphgen: A scalable approach to domain-agnostic labeled graph generation, in: Proceedings of the Web Conference 2020, 2020, pp. 1253–1263.

[38] D. Bacciu, M. Podda, Graphgen-redux: A fast and lightweight recurrent model for labeled graph generation, in: 2021 International Joint Conference on Neural Networks, IJCNN, IEEE, 2021, pp. 1–8.

[39] R. Gómez-Bombarelli, J.N. Wei, D. Duvenaud, J.M. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T.D. Hirzel, R.P. Adams, A. Aspuru-Guzik, Automatic chemical design using a data-driven continuous representation of molecules, ACS Central Sci. 4 (2) (2018) 268–276.

[40] W. Jin, R. Barzilay, T. Jaakkola, Junction tree variational autoencoder for molecular graph generation, in: International Conference on Machine Learning, PMLR, 2018, pp. 2323–2332.

[41] C. Yang, P. Zhuang, W. Shi, A. Luu, P. Li, Conditional structure generation through graph variational generative adversarial nets, Adv. Neural Inf. Process. Syst. 32 (2019).

[42] M. Zhang, S. Jiang, Z. Cui, R. Garnett, Y. Chen, D-vae: A variational autoencoder for directed acyclic graphs, Adv. Neural Inf. Process. Syst. 32 (2019).

[43] Y. Du, X. Guo, A. Shehu, L. Zhao, Interpretable molecule generation via disentanglement learning, in: Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics, 2020, pp. 1–8.

[44] X. Guo, Y. Du, L. Zhao, Property controllable variational autoencoder via invertible mutual dependence, in: International Conference on Learning Representations, 2020.

[45] X. Guo, L. Zhao, Z. Qin, L. Wu, A. Shehu, Y. Ye, Interpretable deep graph generation with node-edge co-disentanglement, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 1697–1707.

[46] R.-R. Griffiths, J.M. Hernández-Lobato, Constrained Bayesian optimization for automatic chemical design using variational autoencoders, Chem. Sci. 11 (2) (2020) 577–586.

[47] B. Samanta, A. De, G. Jana, V. Gómez, P. Chattaraj, N. Ganguly, M. Gomez-Rodriguez, Nevae: A deep generative model for molecular graphs, J. Mach. Learn. Res. 21 (114) (2020) 1–33.

[48] J. Li, J. Yu, J. Li, H. Zhang, K. Zhao, Y. Rong, H. Cheng, J. Huang, Dirichlet graph variational autoencoder, Adv. Neural Inf. Process. Syst. 33 (2020) 5274–5283.

[49] Y. Du, Y. Wang, F. Alam, Y. Lu, X. Guo, L. Zhao, A. Shehu, Deep latent-variable models for controllable molecule generation, in: 2021 IEEE International Conference on Bioinformatics and Biomedicine, BIBM, IEEE, 2021, pp. 372–375.

[50] X. Guo, Y. Du, S. Tadepalli, L. Zhao, A. Shehu, Generating tertiary protein structures via interpretable graph variational autoencoders, Bioinform. Adv. 1 (1) (2021) vbab036.

[51] X. Guo, Y. Du, L. Zhao, Disentangled deep generative model for spatial networks, in: ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2021.

[52] Y. Du, X. Guo, A. Shehu, L. Zhao, Interpretable molecular graph generation via monotonic constraints, in: Proceedings of the 2022 SIAM International Conference on Data Mining, SDM, SIAM, 2022, pp. 73–81.

[53] Y. Du, X. Guo, H. Cao, Y. Ye, L. Zhao, Disentangled spatiotemporal graph generative models, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, (6) 2022, pp. 6541–6549.

[54] K. Madhawa, K. Ishiguro, K. Nakago, M. Abe, GraphNVP: An invertible flow-based model for generating molecular graphs, 2019.

[55] C. Zang, F. Wang, Moflow: an invertible flow model for generating molecular graphs, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2020, pp. 617–626.

[56] Y. Luo, K. Yan, S. Ji, Graphdf: A discrete flow model for molecular graph generation, in: International Conference on Machine Learning, PMLR, 2021, pp. 7192–7203.

[57] W. Jin, K. Yang, R. Barzilay, T. Jaakkola, Learning multimodal graph-to-graph translation for molecule optimization, in: International Conference on Learning Representations, 2018.

[58] S. Fan, B. Huang, Conditional labeled graph generation with GANs, in: Proc. ICLR Workshop Represent. Learn. Graphs Manifolds, 2019.

[59] Ł. Maziarka, A. Pocha, J. Kaczmarczyk, K. Rataj, T. Danel, M. Warchoł, Mol-cyclegan: a generative model for molecular optimization, J. Cheminformatics 12 (1) (2020) 2.

[60] S. Yang, J. Liu, K. Wu, M. Li, Learning to generate time series conditioned graphs with generative adversarial nets, 2020, arXiv preprint arXiv:2003.01436.

[61] S. Pölsterl, C. Wachinger, Adversarial learned molecular graph inference and generation, in: Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part II, Springer, 2021, pp. 173–189.

[62] H. Lin, Y. Huang, M. Liu, X. Li, S. Ji, S.Z. Li, Diffbp: Generative diffusion of 3d molecules for target protein binding, 2022, arXiv preprint arXiv:2211.11214.

[63] A. Schneuing, Y. Du, C. Harris, A. Jamasb, I. Igashov, W. Du, T. Blundell, P. Lió, C. Gomes, M. Welling, et al., Structure-based drug design with equivariant diffusion models, 2022, arXiv preprint arXiv:2210.13695.

[64] S. Luo, Y. Su, X. Peng, S. Wang, J. Peng, J. Ma, Antigen-specific antibody design and optimization with diffusion-based generative models for protein structures, Adv. Neural Inf. Process. Syst. 35 (2022) 9754–9767.

[65] N. Anand, T. Achim, Protein structure and sequence generation with equivariant denoising diffusion probabilistic models, 2022, arXiv preprint arXiv:2205.15019.

[66] C. Shi, C. Wang, J. Lu, B. Zhong, J. Tang, Protein sequence and structure co-design with equivariant translation, 2022.

[67] L. Huang, H. Zhang, T. Xu, K.-C. Wong, Mdm: Molecular diffusion model for 3d molecule generation, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, (4) 2023, pp. 5105–5112.

[68] C. Vignac, I. Krawczuk, A. Siraudin, B. Wang, V. Cevher, P. Frossard, Digress: Discrete denoising diffusion for graph generation, in: Proceedings of the 11th International Conference on Learning Representations, 2023.

[69] J. Guan, W.W. Qian, X. Peng, Y. Su, J. Peng, J. Ma, 3D equivariant diffusion for target-aware molecule generation and affinity prediction, 2023.

[70] G. Corso, B. Jing, R. Barzilay, T. Jaakkola, et al., DiffDock: Diffusion steps, twists, and turns for molecular docking, in: International Conference on Learning Representations (ICLR 2023), 2023.

[71] N.T. Runcie, A.S. Mey, Silvr: Guided diffusion for molecule generation, J. Chem. Inf. Model. 63 (19) (2023) 5996–6005.

[72] J. Yim, Diffusion Probabilistic Modeling of Protein Backbones in 3D for the Motif-Scaffolding problem (Ph.D. thesis), Massachusetts Institute of Technology, 2023.

[73] B. Qiang, Y. Song, M. Xu, J. Gong, B. Gao, H. Zhou, W.-Y. Ma, Y. Lan, Coarse-to-fine: a hierarchical diffusion model for molecule generation in 3d, in: International Conference on Machine Learning, PMLR, 2023, pp. 28277–28299.

[74] C.A. Grambow, H. Weir, N.L. Diamant, A.M. Tseng, T. Biancalani, G. Scalia, K.V. Chuang, RINGER: Rapid conformer generation for macrocycles with sequence-conditioned internal coordinate diffusion, 2023, arXiv preprint arXiv:2305.19800.

[75] M. Xu, A.S. Powers, R.O. Dror, S. Ermon, J. Leskovec, Geometric latent diffusion models for 3d molecule generation, in: International Conference on Machine Learning, PMLR, 2023, pp. 38592–38610.

[76] W. Zhang, X. Wang, J. Smith, J. Eaton, B. Rees, Q. Gu, Diffmol: 3d structured molecule generation with discrete denoising diffusion probabilistic models, in: ICML 2023 Workshop on Structured Probabilistic Inference Backslash and Generative Modeling, 2023.

[77] J. Guan, X. Zhou, Y. Yang, Y. Bao, J. Peng, J. Ma, Q. Liu, L. Wang, Q. Gu, DECOMPDIFF: Diffusion models with decomposed priors for structure-based drug design, Proc. Mach. Learn. Res. 202 (2023) 11827–11846.

[78] M.A. Shabani, S. Hosseini, Y. Furukawa, Housediffusion: Vector floorplan generation via a diffusion model with discrete and continuous denoising, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 5466–5475.

[79] H. Wen, Y. Lin, Y. Xia, H. Wan, Q. Wen, R. Zimmermann, Y. Liang, Diffstg: Probabilistic spatio-temporal graph forecasting with denoising diffusion models, in: Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems, 2023, pp. 1–12.

[80] X. Chen, M. Wu, L. Liu, Edge++: Improved training and sampling of EDGE, in: NeurIPS 2023 Workshop on Synthetic Data Generation with Generative AI.

[81] K.-H. Lee, G.J. Yun, Microstructure reconstruction using diffusion-based generative models, Mech. Adv. Mater. Struct. (2023) 1–19.

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

[83] Z. Sun, Y. Yang, Difusco: Graph-based diffusion solvers for combinatorial optimization, Adv. Neural Inf. Process. Syst. 36 (2023) 3706–3731.

[84] C. Vignac, N. Osman, L. Toni, P. Frossard, Midi: Mixed graph and 3d denoising diffusion for molecule generation, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2023, pp. 560–576.

[85] A. Morehead, J. Cheng, Geometry-complete diffusion for 3D molecule generation and optimization, Commun. Chem. 7 (1) (2024) 150.

[86] L. Huang, T. Xu, Y. Yu, P. Zhao, X. Chen, J. Han, Z. Xie, H. Li, W. Zhong, K.-C. Wong, et al., A dual diffusion model enables 3D molecule generation and lead optimization based on target pockets, Nat. Commun. 15 (1) (2024) 2657.

[87] S. Kim, J. Woo, W.Y. Kim, Diffusion-based generative AI for exploring transition states from 2D molecular graphs, Nat. Commun. 15 (1) (2024) 341.

[88] H. Lin, Y. Huang, O. Zhang, Y. Liu, L. Wu, S. Li, Z. Chen, S.Z. Li, Functional-group-based diffusion for pocket-specific molecule generation and elaboration, Adv. Neural Inf. Process. Syst. 36 (2024).

[89] K. Yi, B. Zhou, Y. Shen, P. Liò, Y. Wang, Graph denoising diffusion for inverse protein folding, Adv. Neural Inf. Process. Syst. 36 (2024).

[90] C. Xu, H. Wang, W. Wang, P. Zheng, H. Chen, Geometric-facilitated denoising diffusion model for 3D molecule generation, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, (1) 2024, pp. 338–346.

[91] C. Hua, S. Luan, M. Xu, Z. Ying, J. Fu, S. Ermon, D. Precup, Mudiff: Unified diffusion for complete molecule generation, in: Learning on Graphs Conference, PMLR, 2024, pp. 1–26.

[92] L. Zhao, X. Ding, L. Akoglu, Pard: Permutation-invariant autoregressive diffusion for graph generation, 2024, arXiv preprint arXiv:2402.03687.

[93] K. Martinkus, J. Ludwiczak, W.-C. Liang, J. Lafrance-Vanasse, I. Hotzel, A. Rajpal, Y. Wu, K. Cho, R. Bonneau, V. Gligorijevic, et al., AbDiffuser: full-atom generation of in-vitro functioning antibodies, Adv. Neural Inf. Process. Syst. 36 (2024).

[94] H. Chen, J. Ding, Y. Li, Y. Wang, X.-P. Zhang, Social physics informed diffusion model for crowd simulation, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, (1) 2024, pp. 474–482.

[95] C. Fu, K. Yan, L. Wang, W.Y. Au, M.C. McThrow, T. Komikado, K. Maruhashi, K. Uchino, X. Qian, S. Ji, A latent diffusion model for protein structure generation, in: Learning on Graphs Conference, PMLR, 2024, pp. 1–17.

[96] R. Jiao, W. Huang, P. Lin, J. Han, P. Chen, Y. Lu, Y. Liu, Crystal structure prediction by joint equivariant diffusion, Adv. Neural Inf. Process. Syst. 36 (2024).

[97] R. Jiao, W. Huang, Y. Liu, D. Zhao, Y. Liu, Space group constrained crystal generation, 2024.

[98] X. Fu, Y. Gao, Y. Wei, Q. Sun, H. Peng, J. Li, L. Xianxian, Hyperbolic geometric latent diffusion model for graph generation, 2024.

[99] A. Klipfel, Y. Fregier, A. Sayede, Z. Bouraoui, Vector field oriented diffusion model for crystal material generation, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, (20) 2024, pp. 22193–22201.

[100] Z. Song, Z. Meng, I. King, A diffusion-based pre-training framework for crystal property prediction, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, (8) 2024, pp. 8993–9001.

[101] H. Park, X. Yan, R. Zhu, E.A. Huerta, S. Chaudhuri, D. Cooper, I. Foster, E. Tajkhorshid, A generative artificial intelligence framework based on a molecular diffusion model for the design of metal-organic frameworks for carbon capture, Commun. Chem. 7 (1) (2024) 21.

[102] J. Lei, C. Deng, W.B. Shen, L.J. Guibas, K. Daniilidis, Nap: Neural 3d articulated object prior, Adv. Neural Inf. Process. Syst. 36 (2024).

[103] R. Yang, Y. Yang, F. Zhou, Q. Sun, Directional diffusion models for graph representation learning, Adv. Neural Inf. Process. Syst. 36 (2024).

[104] A. Bergmeister, K. Martinkus, N. Perraudin, R. Wattenhofer, Efficient and scalable graph generation through iterative local expansion, in: 12th International Conference on Learning Representations, ICLR 2024, OpenReview, 2024.

[105] X. Liu, Y. He, B. Chen, M. Zhou, Advancing graph generation through beta diffusion, in: The Thirteenth International Conference on Learning Representations, 2024.

[106] Z. Li, L. Xia, H. Hua, S. Zhang, S. Wang, C. Huang, DiffGraph: Heterogeneous graph diffusion model, in: Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining, 2025, pp. 40–49.

[107] C. Niu, Y. Song, J. Song, S. Zhao, A. Grover, S. Ermon, Permutation invariant graph generation via score-based generative modeling, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2020, pp. 4474–4484.

[108] C. Shi, S. Luo, M. Xu, J. Tang, Learning gradient fields for molecular conformation generation, in: International Conference on Machine Learning, PMLR, 2021, pp. 9558–9568.

[109] W. Du, H. Zhang, Y. Du, Q. Meng, W. Chen, N. Zheng, B. Shao, T.-Y. Liu, SE (3) equivariant graph neural networks with complete local frames, in: International Conference on Machine Learning, PMLR, 2022, pp. 5583–5608.

[110] S. Luo, C. Shi, M. Xu, J. Tang, Predicting molecular conformation via dynamic graph score matching, Adv. Neural Inf. Process. Syst. 34 (2021) 19784–19795.

[111] J.S. Lee, J. Kim, P.M. Kim, ProteinSGM: Score-based generative modeling for de novo protein design, 2022, BioRxiv, 2022-2007.

[112] L. Wu, C. Gong, X. Liu, M. Ye, Q. Liu, Diffusion-based molecule generation with informative prior bridges, Adv. Neural Inf. Process. Syst. 35 (2022) 36533–36545.

[113] L. Yang, Z. Zhang, W. Zhang, S. Hong, Score-based graph generative modeling with self-guided latent diffusion, 2022.

[114] M. Arts, V. Garcia Satorras, C.-W. Huang, D. Zugner, M. Federici, C. Clementi, F. Noé, R. Pinsler, R. van den Berg, Two for one: Diffusion models and force fields for coarse-grained molecular dynamics, J. Chem. Theory Comput. 19 (18) (2023) 6151–6159.

[115] J. Jo, D. Kim, S.J. Hwang, Graph generation with destination-predicting diffusion mixture, 2023.

[116] P.O. O Pinheiro, J. Rackers, J. Kleinhenz, M. Maser, O. Mahmood, A. Watkins, S. Ra, V. Sresht, S. Saremi, 3D molecule generation by denoising voxel grids, Adv. Neural Inf. Process. Syst. 36 (2024).

[117] F. Bao, M. Zhao, Z. Hao, P. Li, C. Li, J. Zhu, Equivariant energy-guided sde for inverse molecular design, in: The Eleventh International Conference on Learning Representations, 2022.

[118] H. Huang, L. Sun, B. Du, Y. Fu, W. Lv, Graphgdp: Generative diffusion processes for permutation invariant graph generation, in: 2022 IEEE International Conference on Data Mining, ICDM, IEEE, 2022, pp. 201–210.

[119] X. Chen, Y. Li, A. Zhang, L.-p. Liu, Nvdiff: Graph generation through the diffusion of node vectors, 2022, arXiv preprint arXiv:2211.10794.

[120] H. Huang, L. Sun, B. Du, W. Lv, Conditional diffusion based on discrete graph structures for molecular graph generation, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, (4) 2023, pp. 4302–4311.

[121] F. Wu, S.Z. Li, DIFFMD: a geometric diffusion model for molecular dynamics simulations, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, (4) 2023, pp. 5321–5329.

[122] B. Jing, E. Erives, P. Pao-Huang, G. Corso, B. Berger, T.S. Jaakkola, EigenFold: Generative protein structure prediction with diffusion models, in: ICLR 2023-Machine Learning for Drug Discovery Workshop.

[123] H. Huang, L. Sun, B. Du, W. Lv, Learning joint 2d & 3d diffusion models for complete molecule generation, 2023, arXiv preprint arXiv:2305.12347.

[124] X. Han, C. Shan, Y. Shen, C. Xu, H. Yang, X. Li, D. Li, Training-free multi-objective diffusion model for 3D molecule generation, in: The Twelfth International Conference on Learning Representations, 2023.

[125] T. Luo, Z. Mo, S.J. Pan, Fast graph generation via spectral diffusion, IEEE Trans. Pattern Anal. Mach. Intell. (2023).

[126] S. An, H. Lee, J. Jo, S. Lee, S.J. Hwang, Diffusionnag: Taskguided neural architecture generation with diffusion models, 2023, arXiv preprint arXiv:2305.16943.

[127] Y. Qin, H. Wu, W. Ju, X. Luo, M. Zhang, A diffusion model for poi recommendation, ACM Trans. Inf. Syst. 42 (2) (2023) 1–27.

[128] Z. Qiao, W. Nie, A. Vahdat, T.F. Miller III, A. Anandkumar, State-specific protein–ligand complex structure prediction with a multiscale deep generative model, Nat. Mach. Intell. 6 (2) (2024) 195–208.

[129] T. Hsu, B. Sadigh, V. Bulatov, F. Zhou, Score dynamics: Scaling molecular dynamics with picoseconds time steps via conditional diffusion model, J. Chem. Theory Comput. 20 (6) (2024) 2335–2348.

[130] J. Zhu, Z. Gu, J. Pei, L. Lai, DiffBindFR: an SE (3) equivariant network for flexible protein–ligand docking, Chem. Sci. 15 (21) (2024) 7926–7942.

[131] L. Wen, X. Tang, M. Ouyang, X. Shen, J. Yang, D. Zhu, M. Chen, X. Wei, Hyperbolic graph diffusion model, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, (14) 2024, pp. 15823–15831.

[132] Y. Wang, S. Zhang, J. Ye, H. Peng, L. Sun, A mixed-curvature graph diffusion model, in: Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, 2024, pp. 2482–2492.

[133] C. Shi, M. Xu, Z. Zhu, W. Zhang, M. Zhang, J. Tang, Graphaf: a flow-based autoregressive model for molecular graph generation, 2020, arXiv preprint arXiv:2001.09382.

[134] P. Lippe, E. Gavves, Categorical normalizing flows via continuous transformations, 2020, arXiv preprint arXiv:2006.09790.

[135] Z. Chen, Z. Song, Z. Ge, Variational inference over graph: Knowledge representation for deep process data analytics, IEEE Trans. Knowl. Data Eng. (2023).

[136] M. Simonovsky, N. Komodakis, Graphvae: Towards generation of small graphs using variational autoencoders, in: Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I 27, Springer, 2018, pp. 412–422.

[137] M.J. Kusner, B. Paige, J.M. Hernández-Lobato, Grammar variational autoencoder, in: International Conference on Machine Learning, PMLR, 2017, pp. 1945–1954.

[138] A. Bojchevski, O. Shchur, D. Zügner, S. Günnemann, Netgan: Generating graphs via random walks, in: International Conference on Machine Learning, PMLR, 2018, pp. 610–619.

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

[140] K.K. Haefeli, K. Martinkus, N. Perraudin, R. Wattenhofer, Diffusion models for graphs benefit from discrete state spaces, in: The First Learning on Graphs Conference, OpenReview, 2022.

[141] X. Wan, H. Kenlay, B. Ru, A. Blaas, M. Osborne, X. Dong, Attacking graph classification via Bayesian optimisation, in: ICML 2021 Workshop on Adversarial Machine Learning.

[142] F. Guth, S. Coste, V. De Bortoli, S. Mallat, Wavelet score-based generative modeling, Adv. Neural Inf. Process. Syst. 35 (2022) 478–491.

[143] J. Baek, M. Kang, S.J. Hwang, Accurate learning of graph representations with graph multiset pooling, in: International Conference on Learning Representations.

[144] Y. Shou, X. Cao, H. Liu, D. Meng, Masked contrastive graph representation learning for age estimation, Pattern Recognit. 158 (2025) 110974.

[145] T. Meng, Y. Shou, W. Ai, N. Yin, K. Li, Deep imbalanced learning for multimodal emotion recognition in conversations, IEEE Trans. Artif. Intell. (2024).

[146] T. Meng, F. Zhang, Y. Shou, H. Shao, W. Ai, K. Li, Masked graph learning with recurrent alignment for multimodal emotion recognition in conversation, IEEE/ACM Trans. Audio Speech Lang. Process. (2024).

[147] Y. Shou, X. Cao, D. Meng, Spegcl: Self-supervised graph spectrum contrastive learning without positive samples, IEEE Trans. Neural Netw. Learn. Syst. (2025).

[148] Y. Shou, T. Meng, W. Ai, K. Li, Revisiting multi-modal emotion learning with broad state space models and probability-guidance fusion, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2025, pp. 509–525.

[149] H.F. Trotter, On the product of semi-groups of operators, Proc. Amer. Math. Soc. 10 (4) (1959) 545–551.

[150] L. Sun, J. Hu, M. Li, H. Peng, R-ode: Ricci curvature tells when you will be informed, in: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2024, pp. 2594–2598.

[151] L. Sun, J. Hu, S. Zhou, Z. Huang, J. Ye, H. Peng, Z. Yu, P. Yu, Riccinet: Deep clustering via a riemannian generative model, in: Proceedings of the ACM Web Conference 2024, 2024, pp. 4071–4082.

[152] Y. Shou, H. Lan, X. Cao, Contrastive graph representation learning with adversarial cross-view reconstruction and information bottleneck, Neural Netw. 184 (2025) 107094.

[153] Y. Shou, P. Yan, X. Yuan, X. Cao, Q. Zhao, D. Meng, Graph domain adaptation with dual-branch encoder and two-level alignment for whole slide image-based survival prediction, 2024, arXiv preprint arXiv:2411.14001.

[154] W. Ai, F. Zhang, Y. Shou, T. Meng, H. Chen, K. Li, Revisiting multimodal emotion recognition in conversation from the perspective of graph spectrum, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 39, (11) 2025, pp. 11418–11426.

[155] Y. Shou, J. Yao, T. Meng, W. Ai, C. Chen, K. Li, Gsdnet: Revisiting incomplete multimodality-diffusion emotion recognition from the perspective of graph spectrum, in: Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25. International Joint Conferences on Artificial Intelligence Organization, 2025, pp. 6182–6190.

[156] Y. Shou, T. Meng, W. Ai, K. Li, Multimodal large language models meet multimodal emotion recognition and reasoning: A survey, 2025, arXiv preprint arXiv:2509.24322.

[157] J.C. Butcher, A history of runge-kutta methods, Appl. Numer. Math. 20 (3) (1996) 247–260.

[158] P. Erdős, A. Rényi, On the evolution of random graphs, Publ. Math. Inst. Hung. Acad. Sci 5 (1960) 17–61.

[159] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, T. Eliassi-Rad, Collective classification in network data, AI Mag. 29 (3) (2008) 93–93.

[160] R. Ramakrishnan, P.O. Dral, M. Rupp, O.A. Von Lilienfeld, Quantum chemistry structures and properties of 134 kilo molecules, Sci. Data 1 (1) (2014) 1–7.

[161] J.J. Irwin, T. Sterling, M.M. Mysinger, E.S. Bolstad, R.G. Coleman, ZINC: a free tool to discover chemistry for biology, J. Chem. Inf. Model. 52 (7) (2012) 1757–1768.

[162] C. Morris, N.M. Kriege, F. Bause, K. Kersting, P. Mutzel, M. Neumann, Tudataset: A collection of benchmark datasets for learning with graphs, 2020, arXiv preprint arXiv:2007.08663.

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

[164] D.-T. Lee, B.J. Schachter, Two algorithms for constructing a delaunay triangulation, Int. J. Comput. Inf. Sci. 9 (3) (1980) 219–242.

[165] K. Riesen, H. Bunke, IAM graph database repository for graph based pattern recognition and machine learning, in: Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshop, SSPR & SPR 2008, Orlando, USA, December 4-6, 2008. Proceedings, Springer, 2008, pp. 287–297.

[166] A. Feragen, N. Kasenburg, J. Petersen, M. de Bruijne, K. Borgwardt, Scalable kernels for graphs with continuous attributes, Adv. Neural Inf. Process. Syst. 26 (2013).

[167] K.M. Borgwardt, C.S. Ong, S. Schönauer, S. Vishwanathan, A.J. Smola, H.-P. Kriegel, Protein function prediction via graph kernels, Bioinformatics 21, i47–i56.

[168] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, P. Battaglia, Learning deep generative models of graphs, 2018, arXiv preprint arXiv:1803.03324.

[169] Q. Liu, M. Allamanis, M. Brockschmidt, A. Gaunt, Constrained graph variational autoencoders for molecule design, Adv. Neural Inf. Process. Syst. 31 (2018).

[170] T. Ma, J. Chen, C. Xiao, Constrained generation of semantically valid graphs via regularizing variational autoencoders, Adv. Neural Inf. Process. Syst. 31 (2018).

[171] J. You, B. Liu, Z. Ying, V. Pande, J. Leskovec, Graph convolutional policy network for goal-directed molecular graph generation, Adv. Neural Inf. Process. Syst. 31 (2018).

[172] X. Bresson, T. Laurent, A two-step graph convolutional decoder for molecule generation, 2019, arXiv preprint arXiv:1906.03412.

[173] S. Honda, H. Akita, K. Ishiguro, T. Nakanishi, K. Oono, Graph residual flow for molecular graph generation, 2019, arXiv preprint arXiv:1909.13521.

[174] W. Jin, R. Barzilay, T. Jaakkola, Hierarchical generation of molecular graphs using structural motifs, in: International Conference on Machine Learning, PMLR, 2020, pp. 4839–4848.

[175] M. Liu, K. Yan, B. Oztekin, S. Ji, Graphebm: Molecular graph generation with energy-based models, 2021, arXiv preprint arXiv:2102.00546.

[176] J. Liu, A. Kumar, J. Ba, J. Kiros, K. Swersky, Graph normalizing flows, Adv. Neural Inf. Process. Syst. 32 (2019).

[177] Y. Ommi, M. Yousefabadi, F. Faez, A. Sabour, M. Soleymani Baghshah, H.R. Rabiee, Ccgg: A deep autoregressive model for class-conditional graph generation, in: Companion Proceedings of the Web Conference 2022, 2022, pp. 1092–1098.

[178] L. Kong, J. Cui, H. Sun, Y. Zhuang, B.A. Prakash, C. Zhang, Autoregressive diffusion model for graph generation, in: International Conference on Machine Learning, PMLR, 2023, pp. 17391–17408.

[179] D. Zhang, N. Malkin, Z. Liu, A. Volokhova, A. Courville, Y. Bengio, Generative flow networks for discrete probabilistic modeling, in: International Conference on Machine Learning, PMLR, 2022, pp. 26412–26428.

[180] Z. Mo, T. Luo, S.J. Pan, Graph principal flow network for conditional graph generation, in: Proceedings of the ACM on Web Conference 2024, 2024, pp. 768–779.

[181] R. Liao, Y. Li, Y. Song, S. Wang, W. Hamilton, D.K. Duvenaud, R. Urtasun, R. Zemel, Efficient graph generation with graph recurrent attention networks, Adv. Neural Inf. Process. Syst. 32 (2019).

[182] B. Jing, G. Corso, R. Berlinghieri, T. Jaakkola, Subspace diffusion generative models, in: European Conference on Computer Vision, Springer, 2022, pp. 274–289.

[183] I. Krawczuk, P. Abranches, A. Loukas, V. Cevher, GG-GAN: A geometric graph generative adversarial network, 2021.

[184] Y. Hou, J.-D. Park, W.-Y. Shin, Collaborative filtering based on diffusion models: Unveiling the potential of high-order connectivity, in: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2024, pp. 1360–1369.

[185] D. Lilienthal, P. Mello, M. Eirinaki, S. Tiomkin, Multi-resolution diffusion for privacy-sensitive recommender systems, IEEE Access (2024).

[186] R. Chen, J. Fan, M. Wu, R. Cheng, J. Song, G-diff: A graph-based decoding network for diffusion recommender model, IEEE Trans. Neural Netw. Learn. Syst. (2024).

[187] H. Du, H. Yuan, Z. Huang, P. Zhao, X. Zhou, Sequential recommendation with diffusion models, 2023, arXiv preprint arXiv:2304.04541.

[188] J.L. Watson, D. Juergens, N.R. Bennett, B.L. Trippe, J. Yim, H.E. Eisenach, W. Ahern, A.J. Borst, R.J. Ragotte, L.F. Milles, et al., Broadly applicable and accurate protein design by integrating structure prediction networks and diffusion generative models, 2022, pp. 1–54, BioRxiv.

[189] S. Alamdari, N. Thakkar, R. van den Berg, A. Lu, N. Fusi, A. Amini, K. Yang, Protein generation with evolutionary diffusion: sequence is all you need, in: NeurIPS 2023 Generative AI and Biology (GenBio) Workshop, 2023.

[190] S. Kapur, E. Jenner, S. Russell, Diffusion on syntax trees for program synthesis, 2024, arXiv preprint arXiv:2405.20519.

[191] Q. Yan, Z. Liang, Y. Song, R. Liao, L. Wang, SwinGNN: Rethinking permutation invariance in diffusion models for graph generation, Trans. Mach. Learn. Res..

[192] N. Laabid, S. Rissanen, M. Heinonen, A. Solin, V. Garg, Alignment is key for applying diffusion models to retrosynthesis, 2024, arXiv preprint arXiv:2405.17656.

[193] J. Abramson, J. Adler, J. Dunger, R. Evans, T. Green, A. Pritzel, O. Ronneberger, L. Willmore, A.J. Ballard, J. Bambrick, et al., Accurate structure prediction of biomolecular interactions with AlphaFold 3, Nature 630 (8016) (2024) 493–500.

[194] W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, J. Leskovec, Strategies for pre-training graph neural networks, in: International Conference on Learning Representations, 2020.

[195] M. Liu, Z. Fang, Z. Zhang, M. Gu, S. Zhou, X. Wang, J. Bu, Rethinking propagation for unsupervised graph domain adaptation, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, (12) 2024, pp. 13963–13971.

[196] K. Xiao, J. Cao, Z. Zeng, W.-K. Ling, Graph-based active learning with uncertainty and representativeness for industrial anomaly detection, IEEE Trans. Instrum. Meas. 72 (2023) 1–14.