SparseDiff: Sparse Discrete Diffusion for Scalable Graph Generation

Anonymous authors
Paper under double-blind review

Abstract

Graph generative models encounter significant scaling challenges due to the need to predict the presence or type of edges for every node pair, resulting in quadratic complexity. While some models attempt to support large graph generation, they often impose restrictive assumptions, such as enforcing cluster or hierarchical structures, which can limit generalizability and result in unstable generation quality across various graph types. To address this, we introduce SparseDiff, a novel diffusion framework that leverages the inherent sparsity in large graphs - a highly relaxed assumption that enables efficient sparse modeling without sacrificing generation quality for different datasets. Based on that, SparseDiff reduces the complexity of the three core components in graph diffusion models. It first introduces an efficient noising process that samples sparse noisy graphs with linear complexity relative to the number of edges. During training, SparseDiff combines query edge-based random attention with edge-based graph attention mechanisms, matching graph transformer performance while reducing space complexity. Finally, for inference, at each denoising step, SparseDiff maintains sparsity by incrementally reconstructing the adjacency matrix via adding edge subsets defined by query edges. SparseDiff achieves state-of-the-art results on both small and large datasets, showing its robustness across varying graph sizes and its scalability. Additionally, it ensures faster convergence for large graphs, achieving a fourfold speedup on the large-scale Ego dataset compared to dense models. SparseDiff's efficiency, combined with its effective control over space complexity, positions it as a powerful solution for scaling applications involving large graphs. [1]

1 Introduction

Graph generation plays a pivotal role in various fields, such as molecular chemistry (Vignac et al., 2023b), neural architecture search (Asthana et al., 2024) and social network analysis (Schweimer et al., 2022), for its ability to model complex relationships and create realistic structured data. Over the past decades, random graph models have played a foundational role in graph generation (Erdős et al., 1960; Barabási, 2013). However, their limitations in capturing complex dependencies in real-world data have shifted research toward machine learning based graph generative models. Traditional frameworks like generative adversarial networks (De Cao & Kipf, 2018) and variational autoencoders (Simonovsky & Komodakis, 2018) have primarily focused on small graphs. Recently, denoising diffusion models (Jo et al., 2022; Niu et al., 2020), especially those employing discrete modeling to better capture graph structure (Vignac et al., 2023a), have emerged, setting new benchmarks in graph generation tasks and improving scalability by generating graphs with up to 200 nodes. Nonetheless, scaling to even larger graphs remains challenging, restricting application in fields such as protein generation (Yim et al., 2023), histopathology (Madeira et al., 2023), transportation (Rong et al., 2023) or anomaly detection in financial systems (Li et al., 2023).

The primary complexity constraint in these diffusion models stems from the need to predict interactions for all node pairs, naturally leading to the use of full attention layers and resulting in quadratic computational complexity with respect to the number of nodes. Efforts to mitigate this limitation include strategies such

---

[1] Codes available at `https://anonymous.4open.science/r/SparseDiff-18EE/`.

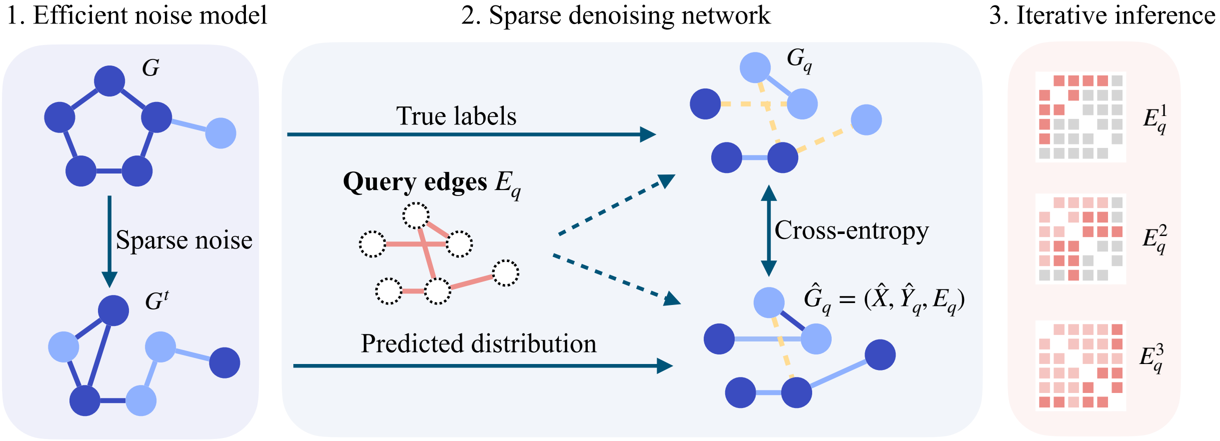


Figure 1: SparseDiff employs three key components for efficiency: (1) a noise model that constructs a sparse noisy graph G^t with linear space complexity relative to the existing edges, (2) a denoising network trained over all nodes and a sparse subset of query edges, $\boldsymbol{E}_q$, which serve a role similar to random batches of all node pairs in stochastic gradient descent, and (3) an iterative inference step that, at each time step t, progressively fills the adjacency matrix using sparse inputs given by $\boldsymbol{E}_q^i$.

as imposing node ordering (Dai et al., 2020), assembling sub-graphs (Limnios et al., 2023), hierarchical generation (Karami, 2023; Jang et al., 2023; Bergmeister et al., 2023), and conditioning on a sampled degree distribution (Chen et al., 2023). While these approaches are designed specifically for large graphs, they rely on additional assumptions such as cluster structures or dependencies on node degrees which can hinder their generalizability. For instance, their suboptimal performance in applications involving smaller graphs, such as molecule generation (Chen et al., 2023; Kong et al., 2023), further raises concerns about the robustness of these assumptions.

Motivated by these limitations and recognizing that most real-world graphs are inherently sparse — for instance, cells in digital pathology connect only with immediate neighbors (Madeira et al., 2023), and connections in social networks are sparse (McCallum et al., 2000) — we propose SparseDiff. SparseDiff is a discrete diffusion framework that leverages graph sparsity without additional assumptions. Unlike previous diffusion models that encode all node pairs, SparseDiff employs a sparse edge list representation, avoiding the quadratic space complexity of adjacency matrices. This enables an efficient realization of the three core components of diffusion models (detailed in Section 2.1) as illustrated in Fig. 1: (1) the noise model to generate noisy data, (2) training of the denoising network, and (3) the inference process.

First, the *efficient noise model* (Section 3.1) preserves the sparsity of noisy graphs throughout the forward trajectory, supported by theoretical guarantees, and also enables efficient sampling with linear space complexity proportional to the number of edges. For denoising, the *sparse denoising neural network* (Section 3.2) applies attention to existing edges, maintaining the computational efficiency of message-passing layers. Crucially, it also incorporates a random subset of node pairs, referred to as query edges, to sparsely approximate full attention. This design bridges the gap between message-passing and full attention, providing a scalable alternative that retains rich interactions granted by full attention mechanisms. The model is trained on random query edges at each step, where query edges serve similarly to random batches in stochastic gradient descent (SGD), enabling efficient training with controlled space complexity while approximately optimizing the same objective. Lastly, SparseDiff introduces an *iterative inference* (Section 3.3) procedure that, at each diffusion step, incrementally fills the dense adjacency matrix with edge subsets defined by query edges, maintaining the same sparsity pattern as in training.

Our experiments demonstrate that SparseDiff consistently achieves state-of-the-art performance on both small graphs, including those with complex priors like molecular graphs, and large graph datasets, including those with over 2,000 nodes. Specifically, SparseDiff outperforms both dense models like SPECTRE (Martinkus et al., 2022) and DiGress (Vignac et al., 2023a), as well as other scalable models like EDGE

(Chen et al., 2023) and HGGT (Jang et al., 2023). Moreover, SparseDiff converges four times faster than dense diffusion models on large graphs, such as social networks (Ego). These results empirically demonstrate the effectiveness of our sparse modeling approach and confirm its robustness through consistently high performance across a diverse range of datasets. In summary, SparseDiff is able to match the performance of dense diffusion models without restrictive assumptions, pioneering the use of sparse representations in graph diffusion and distinguishing it from hierarchical and autoregressive methods.

2 Related Work

2.1 Denoising Diffusion Models for Graphs

Diffusion models have gained increasing popularity due to their impressive performance across various generative tasks in fields such as computer vision (Dhariwal & Nichol, 2021; Ho et al., 2022), protein generation (Baek et al., 2021; Ingraham et al., 2022), and audio synthesis (Kong et al., 2020). These models are characterized by three core components. The first is a Markovian noise model, which progressively corrupts a data point x to a noisy sample z^t over iterative steps t from 1 to T, until it conforms to a simple predefined prior distribution at step T. The second component is a denoising network, parametrized by θ, which is trained to restore the corrupted data back to its less noisy state. Typically, this network aims to predict the original data x given the noisy sample x_t. The third component is the reverse process for data generation, where a fully noisy data point z^T is first sampled from a prior distribution. The denoising network then operates at each time step $t \in [T, ..., 1]$ to predict the less noisy distribution according to $p_\theta(z^{t-1}|z^t) = \int_x q(z^{t-1}|z^t, x)\, dp_\theta(x|z^t)$, from which a new data point z^{t-1} is sampled. While this integral is generally difficult to evaluate, two prominent frameworks, Gaussian diffusion (Ho et al., 2020) and discrete diffusion (Austin et al., 2021), facilitate its efficient computation.

Initial graph diffusion models employed Gaussian noise directly on adjacency matrices (Niu et al., 2020; Jo et al., 2022). These models use a graph attention network to regress the added noise ϵ, where $\epsilon = z^t - z$, effectively regressing the noise up to an affine transformation, akin to regressing the discrete clean graph. To preserve the inherent discreteness of graphical data, subsequent models (Vignac et al., 2023a; Haefeli et al., 2022) have leveraged discrete diffusion, reformulating graph generation as a series of classification tasks, and achieving top-tier results. However, these models require predictions for all pairs of nodes, which implies a quadratic space complexity and thus restricts their scalability.

2.2 Scalable Graph Generation

Efforts to enhance the scalability of graph generative models mainly follow two paradigms: hierarchical refinement and subgraph aggregation. The hierarchical refinement approach initially generates a low-resolution graph, which undergoes successive refinements for enhanced detail (Yang et al., 2021; Karami, 2023). For instance, the HGGT model (Jang et al., 2023) employs a hierarchical K^2-tree representation. For molecule generation, fragment-based models (Jin et al., 2018; 2020; Maziarz et al., 2022) adeptly assemble compounds using pre-defined molecular fragments. Recently, Bergmeister et al. (2023) proposes a hierarchical diffusion model based on spectrum-preserving local expansion algorithms, enabling the generation of non-attributed large graphs. On the other side, the subgraph aggregation approach divides larger graphs into smaller subgraphs, which are subsequently combined. For instance, SnapButton (Yang et al., 2021) enhances autoregressive models (Liu et al., 2018; Liao et al., 2019; Mercado et al., 2021) by merging subgraphs, and SaGess (Limnios et al., 2023) trains a dense diffusion model to generate subgraphs sampled from a large graph that are then merged.

Additionally, some approaches predict all node pairs in an auto-regressive manner. Kong et al. (2023) integrated diffusion with autoregressive models, suggesting learning the node ordering, which is theoretically as difficult as isomorphism testing. Alternatively, EDGE (Chen et al., 2023) uses absorbing states to create sparse diffusion by first generating a node degree distribution $\boldsymbol{d}^0$ and gradually constructing the adjacency matrix $\boldsymbol{A}$ based on node degree changes during inference. While this factorization is universally applicable, the feasibility of learning the conditional distribution $p_\theta(\boldsymbol{A}|\boldsymbol{d}^0)$ remains uncertain, as not all degree distributions can be achieved by undirected graphs. Besides, while latent diffusion is commonly used for scalability in vision tasks, applying it to graphs is more challenging due to permutation equivariance, where the decoded graph can appear in any permutation, necessitating graph matching. Current latent graph diffusion models either use predefined orderings (Evdaimon et al., 2024), or perform diffusion on node features followed by link prediction (Yang et al., 2024), with the latter underperforming SparseDiff or other dense models.

Overall, scalable generation models typically either introduce a dependence on node orderings or rely heavily on extra assumptions on data distribution which can hinder their performance for small graph datasets, while latent graph diffusion faces challenges for graph matching. In contrast, the SparseDiff model described in the next section aims at making no assumption besides graph sparsity, while showing competitive performance compared to other sparse and dense models across a wide range of graphs with different sizes.

3 SparseDiff: Sparse Discrete Diffusion for Graph Generation

We now introduce SparseDiff, a Sparse Denoising Diffusion Model that matches the performance of dense models while significantly enhancing the scalability of graph diffusion to graphs with over $2,000$ nodes, a significant improvement over previous dense models limited to around 200 nodes.

Unlike previous graph diffusion models, SparseDiff builds on a sparse representation of graphs. A graph G, consisting of n nodes and m edges, is represented as a triplet $(\boldsymbol{E}, \boldsymbol{X}, \boldsymbol{Y})$. Here, $\boldsymbol{E} \in \mathbb{N}^{2\times m}$ indicates the edge list with the indices of endpoints. Node and edge attributes are considered to be discrete and are encoded in a one-hot format as $\boldsymbol{X} \in \{0,1\}^{n\times a}$ and $\boldsymbol{Y} \in \{0,1\}^{m\times b}$, where a and b are the number of classes, respectively. In particular, non-existing edges are considered an additional edge type, while edges in $\boldsymbol{E}$ are referred to as *existing edges*. This sparse representation is widely supported by standard graph processing packages such as Pytorch Geometric (Fey & Lenssen, 2019).

This work focuses solely on undirected graphs with discrete attributes, although continuous labels can be seamlessly integrated, in a similar way to Vignac et al. (2023b). All considered graphs are free of self-loops. Figure 1 provides an overview of the SparseDiff framework. Further details on the training and sampling processes are provided in Algorithms 1 and 2, respectively. In the following parts, we specifically focus on three critical components of the diffusion model as detailed in Section 2.1: the noise model, the denoising network, and the sampling algorithm.

3.1 Efficient Noise Model

To improve memory efficiency, SparseDiff employs a dedicated noise model that maintains a similar sparsity level of the noisy graph G^t throughout the noising trajectory under theoretical guarantee, as detailed in Section 3.1.1. This model also reduces the space complexity for computing noisy graph distribution from $O(n^2)$ to $O(m)$ as detailed in Section 3.1.2.

3.1.1 Sparse Trajectory

Given that Gaussian noise applied to the adjacency matrix typically results in dense noisy graphs, where all edge entries in the adjacency matrix acquire continuous values (Niu et al., 2020; Jo et al., 2022) — we opt for a discrete diffusion framework. In this framework, we sample a graph structure from the noisy distribution, allowing us to focus only on existing edges within the noisy graph, thereby reducing the number of edges necessary for computation. In the discrete graph diffusion model, the noisy trajectory at each step is defined by $q(G^t|G^{t-1}) = (\boldsymbol{X}\boldsymbol{Q}^t_{\boldsymbol{X}}, \boldsymbol{Y}\boldsymbol{Q}^t_{\boldsymbol{Y}})$, where $\boldsymbol{Q}^t$ represents the Markov transition matrix for that step t, which transforms G^{t-1} into a noisier distribution until reaching G^T, which follows a predefined prior distribution that is easy to sample. Different types of Markov transition matrices are employed to corrupt the graphs into various prior distributions, including uniform distributions, a special absorbing state (Austin et al., 2021), or marginal distributions (Vignac et al., 2023a).

In this work, we employ the marginal transition model, which favors transitions towards the dominant class in the marginal distribution of the data. This strategy is particularly effective for preserving graph sparsity by naturally biasing toward non-existing edges, which is the dominant edge class for large graphs. Formally, considering the marginal distribution vectors $\boldsymbol{p}_{\boldsymbol{X}}$ for node types and $\boldsymbol{p}_{\boldsymbol{E}}$ for edge types, and denoting their transposes by $\boldsymbol{p}'$, the noise level at each step t is regulated by β^t, with $\alpha^t = 1 - \beta^t$. Formally, the marginal transition matrices are defined as follows:

$$\boldsymbol{Q}^t_{\boldsymbol{X}} = \alpha^t \boldsymbol{I} + \beta^t \mathbf{1}_a \boldsymbol{p}'_{\boldsymbol{X}}, \quad \boldsymbol{Q}^t_{\boldsymbol{Y}} = \alpha^t \boldsymbol{I} + \beta^t \mathbf{1}_b \boldsymbol{p}'_{\boldsymbol{Y}}.$$

Here, $\mathbf{1}_a$ and $\mathbf{1}_b$ are column vectors of ones with dimensions equal to the number of classes a for nodes and b for edges. These matrices incorporate a first term, the identity matrix $\boldsymbol{I}$, to preserve the distribution from G^{t-1}, and a second term to introduce noise aligned with the marginal distributions, namely $\boldsymbol{p}_{\boldsymbol{X}}$ and $\boldsymbol{p}_{\boldsymbol{E}}$.

By employing continuous multiplication, we can derive the distribution at step t directly from the initial clean graph using $q(G^t|G) = (\boldsymbol{X}\bar{\boldsymbol{Q}}^t_{\boldsymbol{X}}, \boldsymbol{Y}\bar{\boldsymbol{Q}}^t_{\boldsymbol{Y}})$, facilitating an immediate transition to the noisy state without the need for iterative step-by-step calculations. For instance, the cumulative transition matrix $\bar{\boldsymbol{Q}}^t$ for nodes $\boldsymbol{X}$ is as follows: $\bar{\boldsymbol{Q}}^t_{\boldsymbol{X}} = \boldsymbol{Q}^1_{\boldsymbol{X}} \boldsymbol{Q}^2_{\boldsymbol{X}} \dots \boldsymbol{Q}^t_{\boldsymbol{X}} = \bar{\alpha}^t \boldsymbol{I} + (1 - \bar{\alpha}^t) \mathbf{1}_a \boldsymbol{p}'_{\boldsymbol{X}}$, where $\bar{\alpha}^t = \alpha^1 \alpha^2 \dots \alpha^t$. The parameter $\bar{\alpha}^t$ starts very close to 1 at $\bar{\alpha}^1$ and approaches 0 by $\bar{\alpha}^T$, reflecting a gradual increase in noise influence over diffusion process.

We note that this choice of marginal noise model does not guarantee that the noisy graph is always sparse. However, it is the case with high probability, as stated by the following lemma, which is an application of Desolneux et al. (2008) (detailed in Appendix A).

Lemma 3.1. *(Probability Bound for Sparsity in Noisy Graphs) Consider an undirected graph with n nodes, m edges, and no self-loops. If the edge ratio given by $m/\left(\frac{n(n-1)}{2}\right)$ is denoted as r, and the edge ratio in the noisy graph sampled from the marginal transition noise model is given by r_t, then for n sufficiently large and $r < \frac{1}{4}$, for any $r < k < 1$, we have:*

$$\log(\mathrm{P}[r_t \geq k]) \sim -\frac{n(n-1)}{2} \left(k \log \frac{k}{r} + (1-r) \log \frac{1-k}{1-r} \right) \tag{1}$$

This lemma demonstrates that, in large and sparse graphs, the probability of the edge ratio r_t in the noisy graph exceeding a threshold k (where $k > r$) declines exponentially with graph size. For instance, in graphs with a low edge ratio r and setting the edge threshold at $k = 2r$, the probability that the noisy graph exceeds k edges is approximately $c_1 e^{-c_2 n^2 r}$, where c_1 and c_2 are constants. This probability decreases substantially as the graph size n grows.

3.1.2 Sparse Computation

The second requirement for the noise model is to achieve subquadratic space complexity in computing the noisy distribution. Standard discrete diffusion models encode all edges using $\boldsymbol{Y} \in \mathbb{R}^{n \times n \times b}$ and calculate transition probabilities with $\boldsymbol{Y}\bar{\boldsymbol{Q}}^t_{\boldsymbol{Y}} \in \mathbb{R}^{n \times n \times b}$, incurring $O(n^2)$ space complexity. To enable sparse sampling, we distinguish between existing and non-existing edges, as the latter are the main contributors to quadratic complexity in sparse graphs.

Specifically, we compute $\boldsymbol{Y}\bar{\boldsymbol{Q}}^t_{\boldsymbol{Y}} \in \mathbb{R}^{m \times b}$ for existing edges, sampling their new labels directly from this distribution, while edges transitioning to non-existing are removed to preserve sparsity in G^t. For non-existing edges, which typically drive quadratic complexity, we introduce a novel three-step approach to sample efficiently without dense adjacency matrices. This process includes:

1. Sampling the number of new existing edges emerging from current non-existing edges.
2. Sampling positions for those new edges, uniformly from non-occupied positions (c.f. Fig. 2).
3. Sampling the edge attributes for those new edges from all existing edge types.

Intuitively, this process decomposes a multinomial distribution for m states into a binomial distribution to firstly filter out non-existing edges, followed by a narrower multinomial distribution over $m-1$ states for only existing edges. Specifically, in Step 1, the number of new edges follows a Binomial distribution $\mathcal{B}(\bar{m}_t, q_t)$,

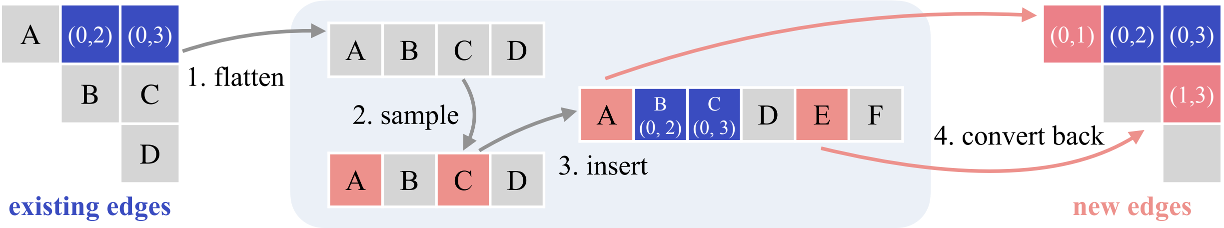


Figure 2: Efficient sampling of new edge positions among non-existing positions is achieved using only the edge list, avoiding the quadratic space complexity of an adjacency matrix. The process involves: 1) Conceptually **flattening** the pair representations into a linear array, 2) **Sampling** uniformly 2 positions from 4 non-occupied positions, selecting A (1st) and C (3rd), 3) **Inserting** offsets of 0 and 2 to the positions of A and C to account for the count of existing edges before the selected positions, resulting in positions A (1st) and E (5th), and 4) **Converting** these positions back to index pairs $(0, 1)$ and $(1, 3)$.

where $\bar{m}_t = \frac{n(n-1)}{2} - m_t$ represents the total number of non-existing edges in G^t, and $q_t = 1 - \boldsymbol{Q}^t[0, 0]$ denotes the transition probability from non-existing to existing edge types, which remains low for sparse graphs. In Step 3, new edge labels are sampled from the Multinomial distribution as specified by $\boldsymbol{Q}^t[0, 1:]$.

However, Step 2—sampling new edge positions—presents additional challenges, as it requires efficiently sampling a certain number of edges from a batch of graphs with varying sizes. Moreover, the sampling must only occur from non-occupied positions, and we must only rely on the edge list rather than the adjacency matrix to avoid high space complexity. As illustrated in Figure 2, we propose an algorithm to efficiently handle this sparse sampling. The method begins by conceptually flattening all vacant positions. Simple random sampling is then performed on these positions using randint to select 2 elements from 4. Crucially, after sampling, the indices have to be adjusted by adding offsets to account for existing edges through a specially designed algorithm, before remapping them back to index pairs.

3.2 Efficient Denoising Neural Network

The traditional model for graph diffusion typically outputs the probabilities for all edges and nodes, which aligns closely with attention layers. These layers explicitly encode edge features with a computational complexity of $O(ln^2 d_e)$, where l is the number of layers and d_e represents the edge feature dimension. In contrast, message-passing methods only account for existing edges, significantly reducing the complexity to $O(lmd_e)$, where m is the number of existing edges.

However, as discussed in Appendix D.6.1, relying solely on message-passing and link prediction using node features tends to perform worse on both small and large datasets. This may be due to the inability to capture long-term and complex interactions between distant nodes, resulting in degraded performance in practice. To address this, we propose bridging the gap between a fully connected attention layer and message-passing with a random attention mechanism. Specifically, we first integrate a convolutional transformer layer (Shi et al., 2020), which leverages attention based only on existing edges. Additionally, we incorporate random attention by sampling query edges randomly from all possible node pairs. By expectation, the model learns the full attention map between nodes, enabling it to capture more complex interactions.

3.2.1 Edge Prediction Module using Sparse Attention

As previously discussed, relying solely on message-passing layers for edge prediction yields limited results. To enhance the performance, we propose a random attention mechanism through a randomly selected subset of node pairs, referred to as query edges. As indicated in Figure 3, the input to the denoising network, referred to as message-passing edges denoted by $(\boldsymbol{E}_m, \boldsymbol{Y}_m)$, thus includes two sets of edges: 'noisy edges' $(\boldsymbol{E}^t, \boldsymbol{Y}^t)$, representing existing edges in the noisy graph G^t, and query edges $(\boldsymbol{E}_q, \boldsymbol{Y}_q)$, which are random edges selected for random attention and for loss computation. Noisy edges preserve the topological information of the noisy graph, while query edges, sampled randomly from all node pairs, include both existing and non-existing edges,

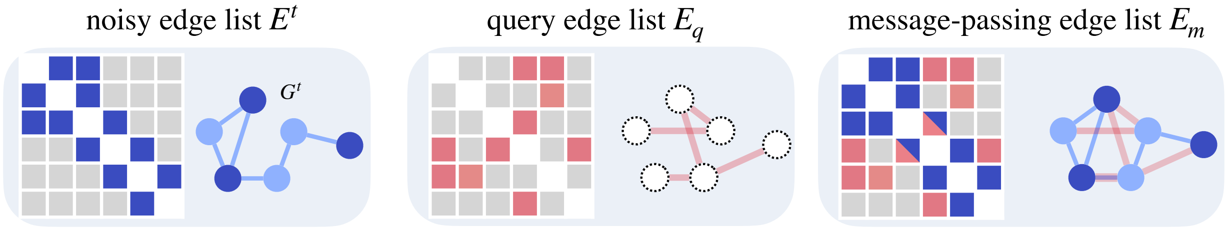


Figure 3: The sparse attention map used in the sparse denoising network is based on the message-passing edge list $\boldsymbol{E}_m$. This edge list consists of noisy edges for graph attention $\boldsymbol{E}^t$ (shown in blue) and uniformly sampled query edges for random attention $\boldsymbol{E}_q$ (shown in red).

enabling the model to predict all edge types. To provide an unbiased estimator of the loss relative to dense diffusion models, query edges are uniformly sampled from all node pairs of each graph.

Additionally, the query edges included in $\boldsymbol{E}_q$ are used for random attention but can also serve as shortcuts within the message-passing network, enabling graph rewiring. This mechanism enhances information propagation and helps mitigate over-squashing issues, as noted in studies such as (Alon & Yahav, 2020; Topping et al., 2021; Di Giovanni et al., 2023).

To control the number of query edges, we define the 'sparsity parameter' λ as the ratio of query edges to all node pairs. Given that query edges may overlap with noisy edges and the number of noisy edges m_t approximates the number of edges m in the clean graph according to Lemma 3.1, the total number of message-passing edges is approximately upper bounded by $(m + \lambda n^2)$. By setting $\lambda = \frac{m}{n^2}$, the computational complexity of SparseDiff practically aligns linearly with the number of edges m in the clean graph G. In our experiments, λ is chosen to enable an effective batch size for training, as detailed in Appendix C.2. This parameter provides flexibility under varying computational conditions and maintains robust performance across different values, as shown in Table 13.

3.2.2 Model Training

Our sparse denoising network adopts a graph transformer architecture featuring normalization, feed-forward, and attention layers (Veličković et al., 2017). It incorporates a sparse attention mechanism for handling sparse data (Shi et al., 2020), and integrates advanced features such as PNA pooling layers (Corso et al., 2020) and FiLM layers (Perez et al., 2018), which are designed to enhance predictive accuracy and effectively manage computational complexity. A detailed discussion of the model architecture is provided in Appendix B.

Training of the network involves predicting query edges $\boldsymbol{E}_q$, and the loss is minimized using the cross-entropy (CE) loss between the predicted distribution $\hat{\boldsymbol{P}}_q^G = (\hat{\boldsymbol{P}}^X, \hat{\boldsymbol{P}}_q^Y)$ and the clean graph G. The loss function is computed as follows:

$$\sum_{1 \le i \le n} \mathrm{CE}(\boldsymbol{X}_i, \hat{\boldsymbol{P}}_i^{\boldsymbol{X}}) + \frac{c}{\lambda} \sum_{(i,j) \in \boldsymbol{E}_q} \mathrm{CE}(\boldsymbol{Y}_{ij}, \hat{\boldsymbol{P}}_{ij}^{\boldsymbol{Y}}), \tag{2}$$

Here, the constant c weights nodes and edges in the loss calculation. It is rescaled by dividing by λ to maintain a consistent edge-to-node weight ratio across different λ values.

Conceptually, the training process of SparseDiff is analogous to stochastic gradient descent compared to gradient descent, with query edges serving as the random batches during training. This design ensures alignment with dense models while maintaining controllable space complexity.

3.3 Iterative Inference

SparseDiff also remains memory-efficient during the inference stage, as visualized in Part (3) of Fig. 1. We start by sampling the number of nodes n of the generated graph from the node distribution of the

Algorithm 1 Sparse training at step t with the sparsity parameter λ (Section 3.1 & 3.2)

```
Given the clean graph $G = (\boldsymbol{E}^0, \boldsymbol{X}^0, \boldsymbol{Y}^0)$;
Sample the noisy graph $G^t = (\boldsymbol{E}^t, \boldsymbol{X}^t, \boldsymbol{Y}^t)$;
Sample query edges $\boldsymbol{E}_q$ of size $\lceil \lambda n^2 \rceil$ ;
$\boldsymbol{E}_m \leftarrow \boldsymbol{E}^t \cup \boldsymbol{E}_q$, $\boldsymbol{Y}_m \leftarrow \boldsymbol{Y}^t \cup \boldsymbol{Y}_q$;          ▷ Construct message-passing edges
$G_m \leftarrow (\boldsymbol{E}_m, \boldsymbol{X}^t, \boldsymbol{Y}_m)$;          ▷ Construct the message-passing graph
$(\hat{\boldsymbol{P}}^{\boldsymbol{X}}, \hat{\boldsymbol{P}}_q^{\boldsymbol{Y}}) = \phi_\theta(G_m, \boldsymbol{E}_q)$;          ▷ Predict the distribution of nodes and query edges
optimizer. step$(\mathrm{CE}(\hat{\boldsymbol{X}}^0, \hat{\boldsymbol{P}}^{\boldsymbol{X}}) + \mathrm{CE}(\boldsymbol{Y}_q^0, \hat{\boldsymbol{P}}_q^{\boldsymbol{Y}}))$;          ▷ Loss calculation
```

Algorithm 2 Iterative inference at step t with the sparsity parameter λ (Section 3.3)

```
Initialize an empty graph $G^{t-1}$ with unlabeled nodes $\boldsymbol{X}^{t-1}$ and no edges;
Randomly divide all node pairs into $K = \lceil \frac{1}{\lambda} \rceil$ equal-sized chunks $\{C_0, \cdots, C_{K-1}\}$;
for k in range($K$) do
    $\boldsymbol{E}_q \leftarrow C_k$;          ▷ Set query edges
    $\boldsymbol{E}_m \leftarrow \boldsymbol{E}^t \cup \boldsymbol{E}_q$;          ▷ Construct message-passing edges and its attributes $\boldsymbol{Y}_m$
    $G_m \leftarrow (\boldsymbol{E}_m, \boldsymbol{X}^t, \boldsymbol{Y}_m)$;          ▷ Construct the message-passing graph
    $(\hat{\boldsymbol{P}^{\boldsymbol{X}}}, \hat{\boldsymbol{P}_q^{\boldsymbol{Y}}}) = \phi_\theta(G_m, \boldsymbol{E}_q)$;          ▷ Predict the distribution of nodes and query edges
    $\hat{\boldsymbol{X}} = \mathrm{Multinomial}(\hat{\boldsymbol{P}}^{\boldsymbol{X}}), \hat{\boldsymbol{Y}}_q = \mathrm{Multinomial}(\hat{\boldsymbol{P}}_q^{\boldsymbol{Y}})$;          ▷ Sample labels
    $\boldsymbol{X}^{t-1} \leftarrow \hat{\boldsymbol{X}}$;          ▷ Assign node new labels
    $\boldsymbol{Y}^{t-1} \leftarrow \boldsymbol{Y}^{t-1} \cup \hat{\boldsymbol{Y}}_q[\hat{\boldsymbol{Y}}_q! = 0], \boldsymbol{E}^{t-1} \leftarrow \boldsymbol{E}^{t-1} \cup \hat{\boldsymbol{E}}_q[\hat{\boldsymbol{Y}}_q! = 0]$;          ▷ Add existing edges
```

training set, which remains constant during the reverse process. Next, we sample a random graph from the prior distribution $G^T \sim \prod_{i=1}^n \mathrm{Cate}(\boldsymbol{p}_{\boldsymbol{X}}) \times \prod_{1 \leq i < j \leq n} \mathrm{Cate}(\boldsymbol{p}_{\boldsymbol{Y}})$, where $\boldsymbol{p}_{\boldsymbol{X}}$ and $\boldsymbol{p}_{\boldsymbol{Y}}$ represent the marginal probabilities of node and edge classes, respectively. The categorical distribution $\mathrm{Cate}(\boldsymbol{p})$ is used for both nodes and edges. The sparse denoising network ϕ_θ is then recursively applied to predict the clean graph from the noisy one. The denoising processes of SparseDiff are further visualized in Fig. 6.

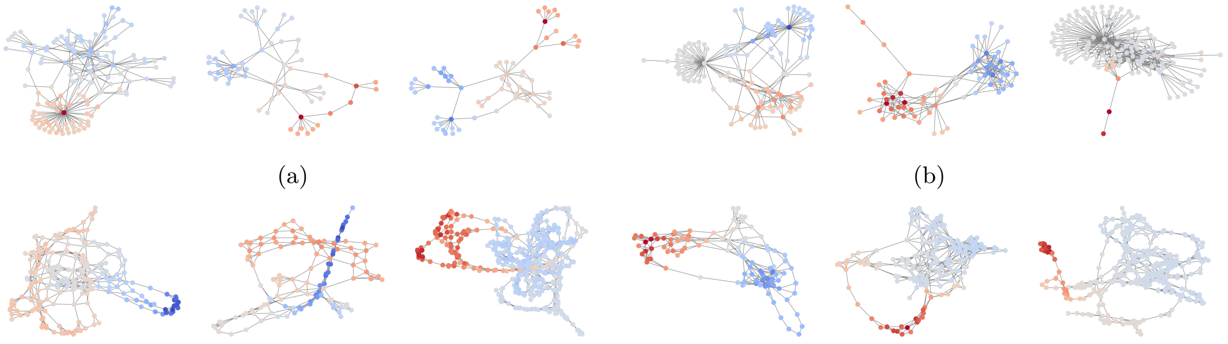
Directly predicting the entire graph at each diffusion step t is impractical due to quadratic memory requirements. Moreover, using dense graphs during inference could lead to a distribution shift compared to the training stage, due to changes in the number of edges used for message-passing. To mitigate this, we implement an iterative procedure to progressively cover all node pairs in G^{t-1}. As detailed in Algorithm 2 and in Part (3) of Fig. 1, we divide all node pairs randomly into $K = \lceil \frac{1}{\lambda} \rceil$ equally-sized sets, representing the query edges for each prediction step[2]. During each iteration k, the noisy graph G^t remains identical, and a message-passing edge list $\boldsymbol{E}_m$ is constructed using noisy edges and query edges from the k^{th} set denoted by $\boldsymbol{E}_q^k$. SparseDiff then predicts the distributions for these query edges, samples labels, and integrates edges classified as existing into G^{t-1}.

This iterative approach allows SparseDiff to maintain favorable memory complexity, albeit at the cost of increased sampling time due to iterations at each diffusion step. However, unlike many scalable models, SparseDiff does not impose additional assumptions about data distribution, such as clustering or degree distribution. Despite the increased sampling time, the generation time for SparseDiff remains efficient due to its ability to utilize larger batch sizes and to accelerate model computations using sparse inputs, as reported in Table 12 of Appendix D.4.

Besides, drawing inspiration from D3PM (Austin et al., 2021) and DDIM (Song et al., 2020), we propose a method to accelerate inference by reducing the inference steps by a factor k. In particular, at each step t, the model predicts $q(G^{t-k}|G^t, G) \propto G^t(Q^t)' \odot G\bar{Q}^{t-k}$ instead of $q(G^{t-1}|G^t, G) \propto G^t(Q^t)' \odot G\bar{Q}^{t-1}$. The results for acceleration are reported in Table 5.

---

[2] When $\frac{n(n-1)}{2}$ is not divisible by K, we adjust by slightly overlapping the last set with the previous one.



(a)

(b)

(c)

(d)

Figure 4: Samples from SparseDiff trained on large graphs. (a) Ego training set (50 to 399 nodes); (b) Generated Ego graphs; (c) Protein training set (100 to 500 nodes); (d) Generated Protein graphs.

4 Experiments

We conducted comprehensive experiments to evaluate the performance of SparseDiff across a diverse set of graphs comparing against a comprehensive range of models. These include GraphRNN (You et al., 2018), GRAN (Liao et al., 2019), GraphNVP (Madhawa et al., 2019), SPECTRE (Martinkus et al., 2022), GDSS (Jo et al., 2022), DiGress (Vignac et al., 2023a), DruM (Jo et al., 2023), and several scalable models such as BiGG (Dai et al., 2020), GraphARM (Kong et al., 2023), EDGE (Chen et al., 2023), HiGen (Karami, 2023), and HGGT (Jang et al., 2023). For clarity, we refer to the local expansion method proposed by Bergmeister et al. (2023) as GraphLE.

To ensure reliability, SparseDiff's performance metrics are reported as mean ± standard deviation, derived from five samples. This approach accounts for the instability caused by the small test set sizes in certain datasets, specifically the SBM and Planar datasets. We **bold** the best-performing method for each metric. The results underscore SparseDiff's significant competitive advantage on datasets containing larger graphs, such as Planar, SBM, Protein, Ego, Facebook and CORA, alongside its state-of-the-art performance on datasets with small molecules, including QM9 and Moses (as detailed in Appendix D.3). Notably, SparseDiff is the only model that demonstrates competitive performance across both large and small graphs, handling both attributed and unattributed graphs effectively.

4.1 Large Graph Generation

Dataset We evaluate SparseDiff on diverse graph datasets to demonstrate its scalability and versatility. First, we test its ability to generate edge-crossing-free planar graphs with 64 nodes. Next, we assess its capacity to generate graphs with 2 to 5 communities using Stochastic Block Model (SBM) graphs, scaling up to 200 nodes—the largest size seen in models like DiGress (Vignac et al., 2023a). We also evaluate Ego and Protein datasets, with graphs up to 500 nodes, representing citation relationships and amino acid interactions within 6 Angstroms. The largest edge ratio for these datasets is 8.8%, confirming their sparsity. Detailed dataset statistics are in Appendix C.2. To further highlight our model's scalability, we include the generation of a large graph with 1,045 nodes in Table 8, and a large graph with 2,485 nodes in Table 9, both thoroughly discussed in Appendix D.1.

Metrics For evaluation, we use maximum mean discrepancy (MMD) metrics, standard in graph generation tasks. We report the validity of SBM graphs as the fraction passing a stochastic block model test, and for Planar graphs, the fraction that are planar and connected. For larger datasets, we also use the Radial Basis Function (RBF) MMD metric to assess fidelity and diversity using a randomly parametrized GNN (Thompson et al., 2022). Since MMD metrics often yield small values that are difficult to compare directly, we report Degree, Cluster, Orbit, Spectre and RBF MMD metrics in units of 10^{-3}, 10^{-2}, 10^{-2}, 10^{-3} and 10^{-2}, respectively. The theoretical optimal metrics, computed with $\mathrm{MMD}(\mathrm{train}, \mathrm{test})^2$, are used as

Table 1: Sample quality on large graphs. The mean ratios to the reference of the Degree, Cluster, Orbit, and Spectre MMD metrics are reported to enable a comprehensive comparison.

Class	Model	Degree(10^{-3})↓	Cluster (10^{-2})↓	Orbit (10^{-2})↓	Spectre (10^{-3})↓	$\overline{\text{Ratio}}$ ↓	RBF (10^{-2})↓
Protein	*Reference*	0.3	0.7	0.3	0.5	1.0	1.4
Dense	GRAN	2.0	4.9	13	5.1	17	–
	DiGress	5.9±0.1	10±1.4	5.1±1.8	2.9±0.5	14±2.3	7.1±1.5
Sparse	DruM	1.9	6.6	3.5	3.0	8.4	–
	BiGG	**1.0**	**2.6**	2.3	4.5	5.9	–
	HiGen	1.2	4.4	2.3	2.5	5.7	–
	GraphLE	3.0	3.1	**0.5**	**1.3**	4.7	–
	SparseDiff	1.5±0.3	3.4±0.3	**0.5**±0.8	1.4±0.2	**3.6**±1.1	**3.8**±0.7
Ego	*Reference*	0.2	0.7	0.7	1.0	1.0	0.9
Dense	DiGress	8.9±1.6	5.4±0.4	3.0±0.3	19±3.2	19±3.1	3.4±0.8
Sparse	EDGE	58	18	5.2	–	107	6.6
	HiGen	47	**0.3**	3.9	–	81	4.5
	SparseDiff	**3.7**±0.4	3.2±0.1	**2.0**±0.4	**5.6**±0.8	**7.9**±0.9	**2.6**±0.3

Table 2: Sample quality on synthetic graphs. The mean ratios to the reference of the Degree, Cluster and Orbit MMD metrics are reported to enable a comprehensive comparison.

Dataset	Stochastic block model					Planar				
Model	Degree↓	Cluster↓	Orbit↓	$\overline{\text{Ratio}}$ ↓	V.U.N.↑	Degree ↓	Cluster ↓	Orbit↓	$\overline{\text{Ratio}}$ ↓	V.U.N.↑
Reference	0.9	3.3	2.6	1.0	100%	0.2	3.1	0.1	1.0	100%
GRAN	5.5	5.8	7.9	3.6	25%	0.7	4.3	**0.1**	2.0	0%
GG-GAN	3.5	7.0	5.9	2.8	25%	63	118	123	528	0%
SPECTRE	1.5	5.2	4.1	1.6	53%	0.5	7.9	**0.1**	2.0	25%
DruM	**0.7**	**4.9**	4.5	3.6	**85%**	0.5	3.5	**0.1**	1.5	90%
HiGen	5.5	5.8	7.9	3.6	–	–	–	–	–	–
GraphLE	12	5.2	6.7	5.8	45%	0.5	6.3	0.2	2.2	**95%**
DiGress	1.7±0.1	5.0±0.1	**3.6**±0.4	**1.6**±0.1	74%±4	0.8±0.0	4.1±0.3	0.5±0.0	**1.2**±0.4	76%±1
SparseDiff	1.6±0.9	5.0±0.1	4.5±0.9	1.7±0.5	75%±10	**0.3**±0.0	**3.2**±0.3	**0.1**±0.1	**1.2**±0.4	85%±9

the reference and represented by a light gray line. Detailed results with higher precision are available in Appendix C.3 for facilitating comparison.

Results The results of large and synthetic graph generation are depicted in Tables 1 and 2, where SparseDiff consistently achieves top scores on aggregated average ratio metrics, trailing DiGress by only 0.1 on SBM. Notably, SparseDiff reaches DiGress's best performance within its variance, indicating that it could achieve this benchmark with an optimal random seed, further underscoring its superior global performance. Specifically, while dense models like DiGress (Vignac et al., 2023a) excel on SBM and planar graphs, they struggle significantly with larger datasets such as Ego and Protein due to their quadratic space complexity. For graphs with over 1,000 nodes, as shown in Tables 8 and 9, these dense models fail immediately due to out-of-memory issues. In contrast, SparseDiff efficiently supports graphs of all sizes, leveraging the controllable sparsity parameter λ. Its performance on mid-sized datasets like SBM and Planar matches both dense and sparse models, due to its robustness without relying on restrictive assumptions. SparseDiff also shows highly competitive with scalable models across various metrics on larger datasets such as Ego and Protein.

4.2 Molecule Generation

Dataset and Metrics Given that our method behaves like dense models in the limit case where $\lambda = 1$, it is expected to align with their performance on small graph datasets. We evaluate our approach using the QM9 and Moses molecular datasets, anticipating its performance comparable to that of dense models. The QM9 dataset (Wu et al., 2018) features molecules with up to 9 heavy atoms, while the Moses benchmark (Polykovskiy et al., 2020), derived from ZINC Clean Leads, includes drug-sized molecules with extensive assessment tools. In QM9, we add formal charges as discrete node features during diffusion, similar to

Table 3: Molecule generation on QM9 with implicit hydrogens. Validity, uniqueness, and connectivity metrics are reported as percentages.

Dense Models					Sparse Models				
Model	Valid ↑	Unique ↑	Conn. ↑	FCD ↓	Model	Valid ↑	Unique ↑	Conn. ↑	FCD ↓
SPECTRE	87.3	35.7	-	-	GraphARM	90.3	-	-	1.22
GraphNVP	83.1	99.2	-	-	EDGE	99.1	**100**	-	0.46
GDSS	95.7	98.5	-	2.9	HGGT	99.2	95.7	-	0.40
DiGress	**99.3**±0.0	95.9±0.2	99.4±0.2	0.15±0.01	SparseDiff	99.2±0.1	96.4±0.1	**99.8**±0.1	**0.12**±0.00

Vignac et al. (2023b), and apply the same to DiGress for consistency. We assess molecular performance by the proportion of connected graphs, validity of the largest connected component verified by RDKit, and uniqueness of over 10,000 molecules. Additionally, we use the Frechet ChemNet Distance (FCD) (Preuer et al., 2018) to measure molecular similarity, excluding 0.96% of invalid molecules for FCD analysis.

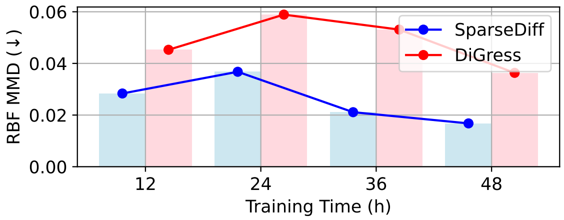
Results Table 3 demonstrates that SparseDiff consistently outperforms other scalable methods on the FCD metric, highlighting its effectiveness for small, structured graphs even without significant sparsity advantages. Additionally, SparseDiff achieves results comparable to the state-of-the-art dense model, DiGress, across other metrics. Additional results for QM9 with explicit hydrogens and the Moses dataset (Tables 10 and 11 in Appendices D.2 and D.3) confirm that SparseDiff matches or exceeds the performance of the state-of-the-art model across various molecular datasets We further validate the robust performance of SparseDiff under different sparsity parameters λ to confirm its stability (Tables 13 in Appendix D.5).

4.3 Efficiency Analysis

Table 4: Convergence comparison of graph diffusion models after 24 hours of training.

Model	Deg.↓	Clust.↓	Orbit↓	$\overline{\text{Ratio}}$ ↓	RBF↓
EDP-GNN	22	36	9.9	59	-
DiGress	4.0	4.9	**3.4**	11	5.6
EDGE	46	18	4.5	87	3.6
GraphLE	58	23	4.2	110	-
SparseDiff	**2.3**	**4.7**	3.6	**7.8**	**3.5**

Figure 5: Convergence comparison between DiGress and SparseDiff.



Training efficiency We compare SparseDiff's performance on the Ego dataset against other diffusion models, including dense models like EDP-GNN and DiGress, and sparse models such as EDGE and GraphLE. Table 4 shows that after 24 hours of training on a V100-32G machine, SparseDiff outperforms all metrics except for a minor increase in Orbit MMD compared to DiGress. The comparison of graph diffusion models' sampling speeds is further presented in Table 12 in Appendix D.4. Figure 5 shows that SparseDiff has a significantly faster convergence speed compared to DiGress, achieving satisfactory results within two days. Notably, a SparseDiff model trained for 12 hours demonstrates an RBF MMD comparable to a DiGress model trained for 48 hours.

Inference efficiency We test the Ego and Planar datasets with different numbers of inference steps (1000, 500, and 200) after training with $T = 1,000$ steps. The results, presented in Table 5, surprisingly demonstrate that even with a 5-fold increase in generation speed (down to 200 steps), our model keeps performing, and consistently outperforms most other dense and scalable models, highlighting its potential for more efficient generation.

Table 5: Inference acceleration results.

Dataset	Steps	Deg.↓	Clust.↓	Orbit ↓	$\overline{\text{Ratio}}$ ↓	RBF↓
Ego	1000	3.6	3.1	1.5	8.2	2.0
	500	2.3	2.9	2.0	6.2	2.1
	200	3.7	3.1	1.6	8.4	2.3
Dataset	Steps	Deg.	Clust.	Orbit	$\overline{\text{Ratio}}$	V.U.N
Planar	1000	0.3	3.2	0.1	1.2	85%
	500	0.3	3.4	0.2	1.5	80%
	200	0.5	3.7	0.4	2.6	69%

5 Conclusion

In this work, we introduced SparseDiff, a scalable discrete denoising diffusion model for graph generation. SparseDiff offers precise control over computational resources by predicting only a subset of edges at each step. Experimental results highlight its superior and robust performance across graphs of varying sizes, making it applicable to tasks such as generating large molecules and community graphs. Additionally, the query edge design, sparse transformer architecture, and iterative sampling procedure of SparseDiff can be seamlessly extended to other iterative graph generation models. Furthermore, the proposed efficient sampling algorithm for non-existing edges is applicable for graph rewiring in various other domains. While SparseDiff meets the demands of most scenarios, its scalability and ability to generate graphs with out-of-distribution node counts could be further enhanced by incorporating a structured hierarchical approach, which is expected for future work.

References

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020.

Rohan Asthana, Joschua Conrad, Youssef Dawoud, Maurits Ortmanns, and Vasileios Belagiannis. Multi-conditioned graph diffusion for neural architecture search. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL `https://openreview.net/forum?id=5VotySkajV`.

Jacob Austin, Daniel Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Minkyung Baek, Frank DiMaio, Ivan Anishchenko, Justas Dauparas, Sergey Ovchinnikov, Gyu Rie Lee, Jue Wang, Qian Cong, Lisa N Kinch, R Dustin Schaeffer, et al. Accurate prediction of protein structures and interactions using a three-track neural network. *Science*, 373(6557):871–876, 2021.

Albert-László Barabási. Network science. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1987), 2013.

Andreas Bergmeister, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Efficient and scalable graph generation through iterative local expansion. *arXiv preprint arXiv:2312.11529*, 2023.

Xiaohui Chen, Jiaxing He, Xu Han, and Li-Ping Liu. Efficient and degree-guided graph generation via discrete diffusion modeling. *arXiv preprint arXiv:2305.04111*, 2023.

Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? In *Advances in neural information processing systems*, volume 33, 2020.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *Advances in Neural Information Processing Systems*, 2020.

Hanjun Dai, Azade Nazi, Yujia Li, Bo Dai, and Dale Schuurmans. Scalable deep generative modeling for sparse graphs. In *International Conference on Machine Learning*. PMLR, 2020.

Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. In *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

Agnés Desolneux, Lionel Moisan, and Jean-Michel Morel. Estimating the binomial tail. *From Gestalt Theory to Image Analysis: A Probabilistic Approach*, pp. 47–63, 2008.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems*, 2021.

Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Lio, and Michael M. Bronstein. On over-squashing in message passing neural networks: The impact of width, depth, and topology. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Paul Erdős, Alfréd Rényi, et al. On the evolution of random graphs. *Publ. math. inst. hung. acad. sci*, 1960.

Iakovos Evdaimon, Giannis Nikolentzos, Michail Chatzianastasis, Hadi Abdine, and Michalis Vazirgiannis. Neural graph generator: Feature-conditioned graph generation using latent diffusion models. *arXiv preprint arXiv:2403.01535*, 2024.

Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

Kilian Konstantin Haefeli, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Diffusion models for graphs benefit from discrete state spaces. *arXiv preprint arXiv:2210.01549*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2020.

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

John Ingraham, Max Baranov, Zak Costello, Vincent Frappier, Ahmed Ismail, Shan Tie, Wujie Wang, Vincent Xue, Fritz Obermeyer, Andrew Beam, et al. Illuminating protein space with a programmable generative model. *bioRxiv*, 2022.

Yunhui Jang, Dongwoo Kim, and Sungsoo Ahn. Hierarchical graph generation with k2-trees. *arXiv preprint arXiv:2305.19125*, 2023.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*. PMLR, 2018.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Hierarchical generation of molecular graphs using structural motifs. In *International Conference on Machine Learning*. PMLR, 2020.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. Score-based generative modeling of graphs via the system of stochastic differential equations. *arXiv preprint arXiv:2202.02514*, 2022.

Jaehyeong Jo, Dongki Kim, and Sung Ju Hwang. Graph generation with destination-predicting diffusion mixture. *arXiv preprint arXiv:2302.03596*, 2023.

Mahdi Karami. Higen: Hierarchical graph generative networks. *arXiv preprint arXiv:2305.19337*, 2023.

Lingkai Kong, Jiaming Cui, Haotian Sun, Yuchen Zhuang, B. Aditya Prakash, and Chao Zhang. Autoregressive diffusion model for graph generation, 2023. URL `https://openreview.net/forum?id=98J48HZXxd5`.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

Xujia Li, Yuan Li, Xueying Mo, Hebing Xiao, Yanyan Shen, and Lei Chen. Diga: Guided diffusion model for graph recovery in anti-money laundering. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023.

Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Charlie Nash, William L. Hamilton, David Duvenaud, Raquel Urtasun, and Richard Zemel. Efficient graph generation with graph recurrent attention networks. In *NeurIPS*, 2019.

Stratis Limnios, Praveen Selvaraj, Mihai Cucuringu, Carsten Maple, Gesine Reinert, and Andrew Elliott. Sagess: Sampling graph denoising diffusion model for scalable graph generation. *arXiv preprint arXiv:2306.16827*, 2023.

Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander Gaunt. Constrained graph variational autoencoders for molecule design. *Advances in neural information processing systems*, 31, 2018.

Manuel Madeira, Dorina Thanou, and Pascal Frossard. Tertiary lymphoid structures generation through graph-based diffusion. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 37–53. Springer, 2023.

Kaushalya Madhawa, Katushiko Ishiguro, Kosuke Nakago, and Motoki Abe. Graphnvp: An invertible flow model for generating molecular graphs. *arXiv preprint arXiv:1905.11600*, 2019.

Karolis Martinkus, Andreas Loukas, Nathanaël Perraudin, and Roger Wattenhofer. Spectre: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. *arXiv preprint arXiv:2204.01613*, 2022.

Krzysztof Maziarz, Henry Richard Jackson-Flux, Pashmina Cameron, Finton Sirockin, Nadine Schneider, Nikolaus Stiefl, Marwin Segler, and Marc Brockschmidt. Learning to extend molecular scaffolds with structural motifs. In *International Conference on Learning Representations (ICLR)*, 2022.

Andrew McCallum, Kamal Nigam, Jason D. M. Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3, 2000.

Rocío Mercado, Tobias Rastemo, Edvard Lindelöf, Günter Klambauer, Ola Engkvist, Hongming Chen, and Esben Jannik Bjerrum. Graph networks for molecular design. *Machine Learning: Science and Technology*, 2021.

Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

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

Kristina Preuer, Philipp Renz, Thomas Unterthiner, Sepp Hochreiter, and Gunter Klambauer. Fréchet chemnet distance: a metric for generative models for molecules in drug discovery. *Journal of chemical information and modeling*, 58(9):1736–1741, 2018.

Can Rong, Jingtao Ding, Zhicheng Liu, and Yong Li. City-wide origin-destination matrix generation via graph denoising diffusion. *arXiv preprint arXiv:2306.04873*, 2023.

Christoph Schweimer, Christine Gfrerer, Florian Lugstein, David Pape, Jan A. Velimsky, Robert Elsässer, and Bernhard C. Geiger. Generating simple directed social network graphs for information spreading. In *Proceedings of the ACM Web Conference 2022*. ACM, 2022. URL `http://dx.doi.org/10.1145/3485447.3512194`.

Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020.

Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *International conference on artificial neural networks*. Springer, 2018.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Rylee Thompson, Boris Knyazev, Elahe Ghalebi, Jungtaek Kim, and Graham W Taylor. On evaluation metrics for graph generative models. *arXiv preprint arXiv:2201.09871*, 2022.

Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv preprint arXiv:2111.14522*, 2021.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. In *The Eleventh International Conference on Learning Representations*, 2023a.

Clément Vignac, Nagham Osman, Laura Toni, and Pascal Frossard. Midi: Mixed graph and 3d denoising diffusion for molecule generation. In *ECML/PKDD*, 2023b.

Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. In *Chem. Sci.* The Royal Society of Chemistry, 2018.

Ling Yang, Zhilin Huang, Zhilong Zhang, Zhongyi Liu, Shenda Hong, Wentao Zhang, Wenming Yang, Bin Cui, and Luxia Zhang. Graphusion: Latent diffusion for graph generation. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

Shuai Yang, Xipeng Shen, and Seung-Hwan Lim. Revisit the scalability of deep auto-regressive models for graph generation. In *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021.

Jason Yim, Brian L Trippe, Valentin De Bortoli, Emile Mathieu, Arnaud Doucet, Regina Barzilay, and Tommi Jaakkola. Se (3) diffusion model with application to protein backbone generation. *arXiv preprint arXiv:2302.02277*, 2023.

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

A Proof of Lemma 3.1

The lemma for a noisy graph with guaranteed sparsity comes directly from the proposition regarding the tail behavior of a binomial distribution (Desolneux et al., 2008) as follows:

Proposition A.1. *(Tail behavior of a binomial distribution)*

Let $X_i, i = 1, ...l$ be independent Bernoulli random variables with parameter $0 < p < \frac{1}{4}$ and let $S_l = \sum_{i=1}^{l} X_i$. Consider a constant $p < r < 1$ or a real function $p < r(l) < 1$. Then according to the Hoeffding inequality, $\mathcal{B}(l, k, p) = \mathbb{P}[S_l \geq k]$ satisfies:

$$-\frac{1}{l} log \mathcal{P}[S_l \geq rl] \geq rlog\frac{r}{p} + (1-p) log\frac{1-r}{1-p} \tag{3}$$

For sparse graphs, the edge ratio r is clearly smaller than $\frac{1}{4}$. Consider then Bernoulli random variables with the parameter r and a noised edge ratio $r < k < 1$ with $l = n(n-1)/2$ (i.e. number of all node pairs in an undirected graph without self-loops) draws, and note the sampled 'existing' edge number $S_{n(n-1)/2}$ as m_t, we have:

$$log(\mathbb{P}[r_t = \frac{m_t}{n(n-1)/2} \geq k]) \leq -\frac{n(n-1)}{2}[k \log\frac{k}{r} + (1-r) \log\frac{1-k}{1-r}] \tag{4}$$

B Model Architecture

Our sparse denoising network adopts the graph transformer architecture Veličković et al. (2017), featuring normalization, feed-forward, and attention layers. To handle sparse data, it incorporates the sparse attention mechanism (Shi et al., 2020) based on weighted message-passing layers and integrates enhancements from Vignac et al. (2023a), such as PNA pooling layers (Corso et al., 2020) and FiLM layers (Perez et al., 2018).

Precisely, we introduce the FiLM layer and the PNA layer inside the model architecture to enhance its performance. Precisely, the FiLM layer is used to combine features at different scales, such as node and edge features. Specifically, given two features $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$, and trainable parameters $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$, the FiLM layer output is calculated as $\text{FiLM}(\boldsymbol{M}_1, \boldsymbol{M}_2) = \boldsymbol{M}_1\boldsymbol{W}_1 + (\boldsymbol{M}_1\boldsymbol{W}_2) \odot \boldsymbol{M}_2 + \boldsymbol{M}_2$. As an illustration, within the convolutional layer, the graph feature $\boldsymbol{M}_2$ is integrated with edge features $\boldsymbol{M}_1$ to enhance predictions. While PNA layer is used as a specialized pooling layer to obtain information from different dimensions of a specific feature. Given the feature X and trainable parameter $\boldsymbol{W}$, $\text{PNA}(\boldsymbol{X}) = \text{cat}(\max(\boldsymbol{X}), \min(\boldsymbol{X}), \text{mean}(\boldsymbol{X}), \text{std}(\boldsymbol{X}))\ \boldsymbol{W}$. For example, node features $\boldsymbol{X}$ are forwarded to a PNA layer for extracting global information across different scales, which is subsequently added to the graph feature to enhance its representation.

Finally, we enrich the message-passing graph with additional encodings, such as the graph Laplacian and cycle counts, to enhance structural and positional information (detailed in Appendix B.1). These encodings can only be computed effectively when the noisy graphs are sparse, which is another significant advantage of discrete diffusion models using marginal transitions. It is worth noting that not all these encodings can be computed in sub-quadratic time. However, in practice, this does not pose an issue as they are not used for back-propagation, which arises as the primary complexity bottleneck during training. For instance, for the large Protein dataset, computing these encodings is five times faster than the forward pass itself. Nevertheless, on very larger graphs, expensive ones should not be computed.

B.1 Additional Encodings

During training, we augment model expressiveness with additional encodings. To make things clear, we divide them into encodings for edges, nodes, and for graphs.

Encoding for graphs We first incorporate graph eigenvalues, known for their critical structural insights, and cycle counts, addressing message-passing neural networks' inability to detect cycles (Chen et al., 2020).

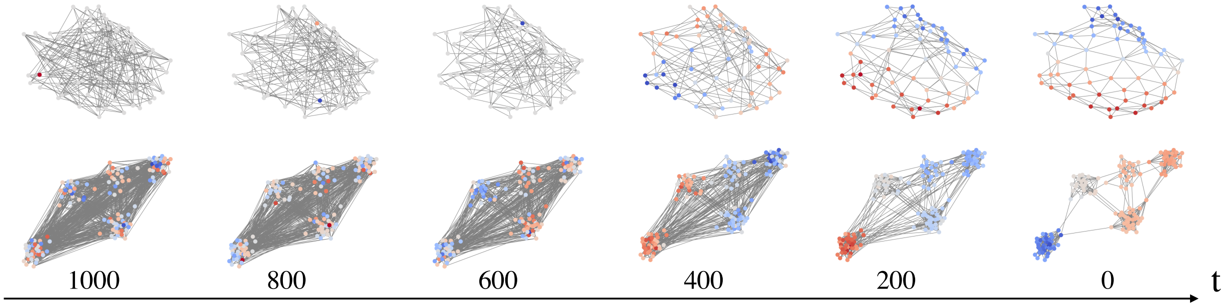


Figure 6: Visualization of iterative generation for Planar and SBM graphs.

The first requires n^3 operations for matrix decomposition, the second requires n^2 for matrix multiplication, but both are optional in our model and do not significantly limit scalability even with graphs up to size 500. In addition to the previously mentioned structural encodings, we integrate the degree distribution to enhance the positional information within the graph input, which is particularly advantageous for graphs with central nodes or multiple communities. Furthermore, for graphs featuring attributed nodes and edges, the inclusion of node type and edge type distributions also provides valuable benefits.

Encoding for nodes At the node level, we use graph eigenvectors, which are fundamental in graph theory, offering valuable insights into centrality, connectivity, and diverse graph properties.

Encoding for edges To aid in edge label prediction, we introduce auxiliary structural encodings related to edges. These include the shortest distance between nodes and the Adamic-Adar index. The former enhances node interactions, while the latter focuses on local neighborhood information. Due to computational constraints, we consider information within a 10-hop radius, categorizing it as local positional information.

Molecular information In molecular datasets, we augment node features by incorporating edge valency and atom weights. Additionally, formal charge information is included as an additional node label for diffusion and denoising during training, as formal charges have been experimentally validated as valuable information (Vignac et al., 2023b).

C Experimental setup

In our experimental setup, we utilize a single V100-32G GPU machine, which is particularly prone to scalability issues, to demonstrate that our method allows users with limited GPU resources to effectively train on larger graphs. Detailed specifications regarding workers, memory allocation, execution time, and optimizers are meticulously indicated in the configuration details provided in our code.

As for dataset splits, we adhere to the framework established by DiGress. Specifically, for the QM9 dataset, we implement a split comprising 100k molecules for training, 20k for validation, and 13k for evaluating likelihood on the test set. For the Planar, SBM, and Protein datasets, employing a seed of 1234, we randomly assign 20% of the graphs to testing, while 80% of the remaining graphs are utilized for training, and 20% for validation. For the Ego dataset, to ensure consistency with previous methods and a fair comparison, we maintain a split of 80% for training and 20% for testing, with 20% of the training set additionally used for validation purposes. All configuration details are comprehensively documented in the code provided.

C.1 MMD metrics

In our research, we carefully select specific metrics tailored to each dataset, with a primary focus on four widely recognized Maximum Mean Discrepancy (MMD) metrics. These metrics use the total variation (TV) distance, as detailed in Martinkus et al. (2022). They encompass node degree (Deg), clustering coefficient (Clus), orbit count (Orb), and graph spectra (Spec). The first three local metrics compare the degree distributions, clustering coefficient distributions, and the occurrence of all 4-node orbits within graphs between the generated and training samples. Additionally, the comparison of graph spectra is realized by computing the eigenvalues of the normalized graph Laplacian, providing complementary insights into the global properties of the graphs.

C.2 Statistics of the datasets

Table 6: Statistics for the datasets employed in our experiments.

Name	Graph number	Node range	Edge range	Edge Ratio (%)	λ (%)
QM9	133,885	[2,9]	[2, 28]	[20, 56]	100
QM9(H)	133,885	[3, 29]	[4, 56]	[7.7, 44]	50
Moses	1,936,962	[8, 27]	[14, 62]	[8.0, 22]	50
Planar	200	[64, 64]	[346, 362]	[8.4, 8.8]	50
SBM	200	[44, 187]	[258, 2258]	[6.0, 17]	25
Ego	757	[50, 399]	[112, 2124]	[1.2, 11]	10
Protein	918	[100, 500]	[372, 3150]	[8.9, 6.7]	10

To provide a more comprehensive overview of the various scales found in 'existing' graph datasets, we present here key statistics for them. These statistics encompass the number of graphs, the range of node numbers, the range of edge numbers, the edge ratio for 'existing' edges, and the sparsity parameter λ used for training, i.e. the proportion of 'existing' edges among all node pairs. In our consideration, we focus on undirected graphs. Therefore, when counting edges between nodes i and j, we include the edge in both directions.

C.3 Raw results

Table 7: Raw results on the SBM, Planar, Protein, and Ego datasets.

Model	Deg. (e-3)↓	Clust. (e-2)↓	Orbit (e-2)↓	Spec. (e-3)↓	FID↓	RBF MMD (e-2)↓
SBM						
Training set	0.85	3.32	2.55	2.74	1.37	3.23
SparseDiff	1.57±0.91	5.04±0.06	4.51±0.90	6.68±2.04	4.55±2.01	4.98±0.06
Planar						
Training set	0.19	3.10	0.05	3.82	1.57	8.89
SparseDiff	0.32±0.01	3.25±0.35	0.09±0.08	6.99±0.92	2.94±3.15	9.84±0.91
Protein						
Training set	0.32	0.68	0.32	0.49	1.36	1.37
SparseDiff	1.45±0.30	3.35±0.33	0.53±0.78	1.35±0.16	5.97±1.07	3.77±0.65
Ego						
Training set	0.16	0.71	0.69	0.98	0.07	0.86
SparseDiff	3.70±0.44	3.18±0.10	1.98±0.42	5.63±0.80	4.84±1.56	2.60±0.31

To ease comparison with other methods, Table 7 provides the raw numbers (not presented as ratios) for the SBM, Planar, Ego, and Protein datasets. Note that this table contains the FID metrics from Thompson et al. (2022), which we did not include in the main text. The reason is that we found this metric to be very brittle, with some evaluations giving a very large value that would dominate the mean results. Besides, we have identified a discrepancy in the Spectre metrics reported in the study by Martinkus et al. (2022) when computed under non-parallel computation. We thus provide the updated values for reference and use the updated value for ratio calculation in Table 2 and in Table 1.

D Additional experiments

D.1 Training with larger graphs

However, using the same graph for both training and evaluation poses potential risks, as high performance metrics could merely reflect overfitting to the training graph. Therefore, we report results on only the two relevant datasets in the appendix.

Table 8: Large graph generation on the Facebook dataset. Triangles and squares are abbreviated as 'tri' and 'squ' in the table, while PLE represents power law exp.

Model	Num Nodes	Num Edges	Num Triangles	Num Squares	Max Deg	Clust	Assort	PLE	CPL
Ref	1045	27,755	446,846	34,098,662	1044	0.57579	-0.02543	1.28698	1.94911
SAGESS-Uni	1043	27,758	429,428	35,261,545	999	0.52098	-0.01607	1.29003	2.00800
SAGESS-RW	1009	27,764	490,844	43,006,252	1001	**0.56138**	-0.02266	1.29398	1.96014
SAGESS-Ego	1005	27,761	515,928	45,421,130	295	0.43074	0.34074	1.29381	2.65926
NetGAN	**1045**	**27,755**	262,574	15,635,626	849	0.39773	-0.01821	1.27429	2.13730
CELL	**1045**	**27,755**	250,968	14,855,676	474	0.30854	0.12788	1.27490	2.38650
DCSBM	1041	27,092	339,448	26,714,948	733	0.37549	0.07125	1.28845	2.33021
SaGess	1043	27,758	429,428	35,261,545	999	0.52098	-0.01607	1.29003	2.00800
SparseDiff	**1045**	27,763	**446,819**	**34,095,513**	**1044**	0.43310	**-0.02536**	**1.28687**	**1.94921**

We first train on the large graph with 1045 nodes from the Facebook dataset, following the SaGess (Limnios et al., 2023) setting. SparseDiff was evaluated using SaGess metrics as a reference. In the provided table, we present SaGess-RW, demonstrating the best results among the three proposed SaGess models. Notably, SaGess generates small graphs and concatenates them to meet the required number of edges, while SparseDiff generates a single large graph based on the specified node count. This explains SparseDiff's advantage in the 'num nodes' metric and SaGess's advantage in the 'num edges' metric. Furthermore, SparseDiff closely aligns with real data statistics, except for the clustering coefficient, showcasing not only its scalability up to 1000 nodes but also its strong performance on such single-graph datasets.

Table 9: Large graph generation on the CORA dataset.

Model	EO	PLE	NTC	CC	CPL	AC
Ref	100	1.885	1	0.090	6.311	-0.071
OPB	10.9	1.852	0.097	0.008	4.476	-0.037
HDOP	0.9	1.849	0.113	0.009	4.770	-0.030
CELL	10.3	1.774	0.009	0.002	5.799	-0.018
CO	9.7	1.776	0.009	0.002	5.653	**0.010**
TSVD	6.7	1.858	0.349	**0.082**	4.908	-0.006
VGAE	1.5	1.717	0.120	0.220	4.934	0.002
GRNN	**0.4**	1.822	0.043	0.011	**6.146**	0.043
EDGE	1.1	1.755	0.446	0.034	4.995	**-0.046**
SparseDiff (pos)	0.3	**1.896**	**1.434**	0.075	4.747	-0.043

We evaluate our model on the CORA dataset (McCallum et al., 2000), as used in EDGE (Chen et al., 2023). The CORA graph consists of 2,485 nodes. Our model is trained with positional encoding for a fair comparison of the edge overlap ratio (EO), with the sparsity parameter set to 0.05. The performance results, after just one day of training, are presented in the table below. As shown, SparseDiff outperforms EDGE in 3 out of 5 metrics and remains competitive in the AC metric, further validating the scalability of our method.

D.2 QM9 with explicit hydrogens

Table 10: Unconditional generation on QM9 with explicit hydrogens. On small graphs such as QM9, sparse models are not beneficial, but SparseDiff still achieves very good performance.

Model	Connected	Valid↑	Unique↑	Atom stable↑	Mol stable↑
DiGress	–	95.4	**97.6**	98.1	79.8
DiGress + charges	**98.6**	97.7	96.9	99.8	97.0
SparseDiff(ours)	98.3±.08	**97.9**±.13	97.4±.10	-	-

We additionally report the results for QM9 with explicit hydrogens in Table 10. Having explicit hydrogens makes the problem more complex because the resulting graphs are larger. We observe that SparseDiff achieves better validity than DiGress and has comparable results on other metrics when both are utilizing charges.

D.3 Moses benchmark evaluation

Table 11: Mean and standard deviation across 5 samplings on the Moses benchmark. SparseDiff has a similar performance to DiGress, despite a shorter training time.

Model	Connected ↑	Valid (%) ↑	Unique (%) ↑	Novel (%) ↑	Filters (%) ↑
GraphINVENT	–	**96.4**	99.8	–	95.0
DiGress	–	85.7	**100.0**	95.0	**97.1**
SparseDiff	94.8±.1	84.7±.2	**100.0**±.0	**95.1**±.1	**97.0**±.2
Model	FCD ↓	SNN (%) ↑	Scaf (%) ↑	Frag (%) ↑	IntDiv (%) ↑
GraphINVENT	1.22	**53.9**	12.7	98.6	**85.7**
DiGress	**1.19**	52.2	**14.8**	99.6	85.3
SparseDiff	1.28±.01	52.2±.0	**15.5**±1.3	**99.8**±.0	85.4±.0
Model	IntDiv2 (%) ↑	logP (e^{-2}) ↓	SA(e^{-2}) ↓	QED (e^{-3}) ↓	Weight (%) ↓
GraphINVENT	**85.1**	**0.67**	4.5	**0.25**	16.1
DiGress	–	3.4	**3.6**	2.91	**1.42**
SparseDiff	84.8±.0	3.0±.3	5.4±.2	1.21±.21	5.58±.15

Moses is an extensive molecular dataset with larger molecular graphs than QM9, offering a much more comprehensive set of metrics. While autoregressive models such as GraphINVENT are recognized for achieving higher validity on this dataset, both SparseDiff and DiGress exhibit advantages across most other metrics. Notably, SparseDiff closely aligns with the results achieved by DiGress, affirming the robustness of our method on complex datasets.

D.4 Sampling Speed Comparison

Table 12: Sampling speed for generating 8 Ego graphs.

Model	EDP-GNN	DiGress	EDGE	GraphLE	SparseDiff	SparseDiff (200 steps)
Time (min)	5	32	2	20	28	5

The speed of generating 8 Ego graphs is demonstrated in Table 12. Notably, for GraphLE, the batch size is constrained to 2 due to its substantial memory requirements. An additional column labeled "SparseDiff

(200 steps)" represents the sampling time after reducing the inference steps from 1,000 to 200 through acceleration strategies. The table illustrates that SparseDiff maintains comparable speed to dense models without significant compromise on space efficiency and can be significantly accelerated during sampling.

D.5 Influence of the sparsity parameter

Table 13: Unconditional generation on QM9 under different sparsity parameters λ.

λ	Valid↑	Unique↑	Connected↑	FCD↓
100%	99.23±0.06	96.37±0.13	99.76±0.06	0.117±0.004
50%	99.12±0.05	96.80±0.18	99.61±0.05	0.107±0.007
25%	99.16±0.06	96.54±0.19	99.59±0.06	0.119±0.006
10%	99.11±0.09	96.89±0.15	99.61±0.01	0.105±0.004

Table 13 above shows the results of unconditional generation on the QM9 dataset under different sparsity parameters (λ). The performance of SparseDiff remains consistent across various λ values, with metrics for connectivity, validity, uniqueness, and Frechet ChemNet Distance (FCD) showing minimal changes from 100% to 10% λ. For example, the validity metric stays between 99.12% and 99.23%, and uniqueness ranges from 96.37% to 96.89%. We remark that, for λ values of 10% and 25%, the model was trained for twice as many epochs since the cross-entropy loss considers fewer edges per epoch, necessitating more epochs for convergence. Despite this, all models exhibit consistent performance across different λ values after convergence, which highlights the robustness and stability of SparseDiff in generating high-quality molecular graphs.

D.6 Ablations

This part presents 2 ablation experiments that motivate our approach. SparseDiff builds upon an experimental observation and a hypothesis. Firstly, our experiments demonstrate that relying solely on node features for link prediction yields unsatisfactory results. This observation encouraged us to design the message-passing graph that contains all edges to be predicted (i.e. query edges) as the message-passing graph to directly obtain their edge features. Secondly, we hypothesized that preserving the same distribution of edge types as observed in dense graphs for loss calculation is advantageous for training. This hypothesis necessitates the sampling of query edges within each graph in a batch of graphs with varying sizes, thereby introducing increased complexity to the algorithm design process.

D.6.1 Link Prediction

Table 14: Influence of including edges features for edge prediction.

Model	Deg ↓	Clus ↓	Orb↓	Spec↓	FID↓	RBF MMD↓
Link Pred	0.0043	0.0721	**0.0275**	0.0344	1.51e6	**0.0315**
SparseDiff	**0.0019**±.00	**0.0537**±.00	0.0299±.00	**0.0050**±.00	**16.1**±12.9	0.0483±.01

Table 15: Influence of including edges features for edge prediction on the QM9 dataset.

Model	Valid↑	Unique↑	Connected↑	FCD↓
Link Pred	98.12	96.25	99.58	0.310
SparseDiff	**99.23**±0.06	**96.37**±0.13	**99.76**±0.06	**0.117**±0.004

In Table 14, the model that does not explicitly incorporate edge features for edge prediction underperforms across all metrics, except for RBF MMD and orbit. Similarly, in Table 15, the link prediction-based method fails to achieve comparable validity, even though QM9 is widely recognized as an easy dataset to learn. Both experiments highlight a subtle yet challenging gap between message-passing and transformer-based architectures for graph generation, as the latter provides richer topological interactions. While, in our case, this ablation proves detrimental, developing a more robust link prediction module could simplify the task to link prediction and significantly reduce space complexity.

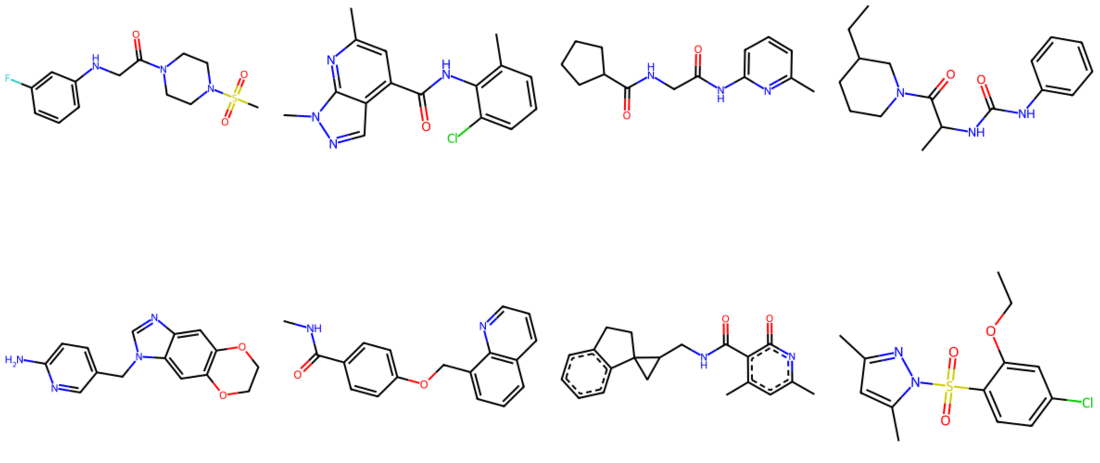
D.6.2 Query edges with proper distribution

Table 16: Influence of edge loss distribution on EGO dataset.

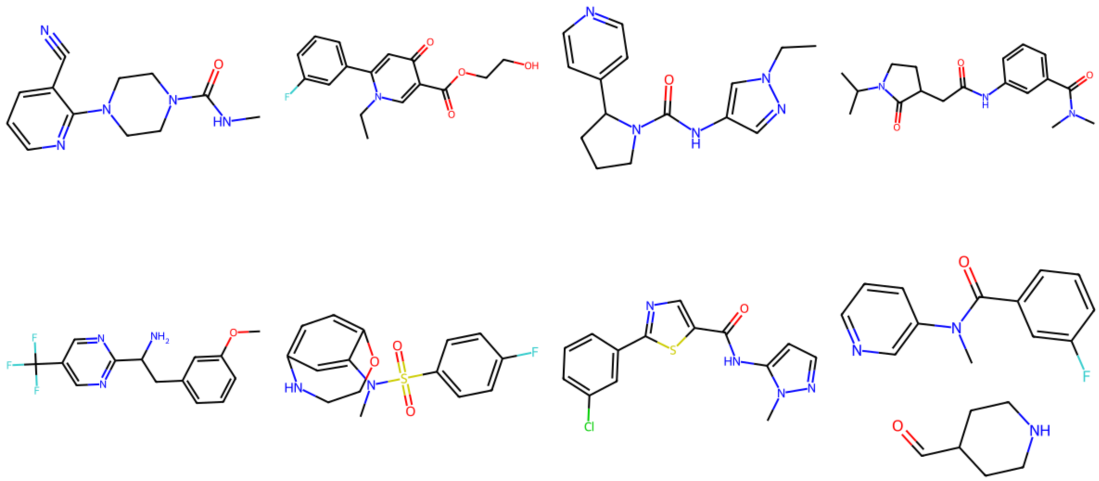
Loss based on	Deg ↓	Clus ↓	Orb↓	Spec↓	FID↓	RBF MMD↓
Comp graph	0.0021	0.0566	**0.0270**	0.0100	28.2	**0.0396**
Query graph	**0.0019**±.00	**0.0537**±.00	0.0299±.00	**0.0050**±.00	**16.1**±12.9	0.0483±.01

In order to emphasize the importance of preserving the edge distribution when computing losses, we conduct an experiment where we assess the performance of a model trained using all message-passing edges as opposed to solely using query edges. The former results in an increased emphasis on ‘existing’ edges during training compared to SparseDiff. Similarly, we use the Ego dataset for initial experiments. Table 16 shows that using edges of the message-passing graph G_m results in worse performance on most of the metrics, which indicates the importance of keeping a balanced edge distribution for loss calculation.

E Visualization

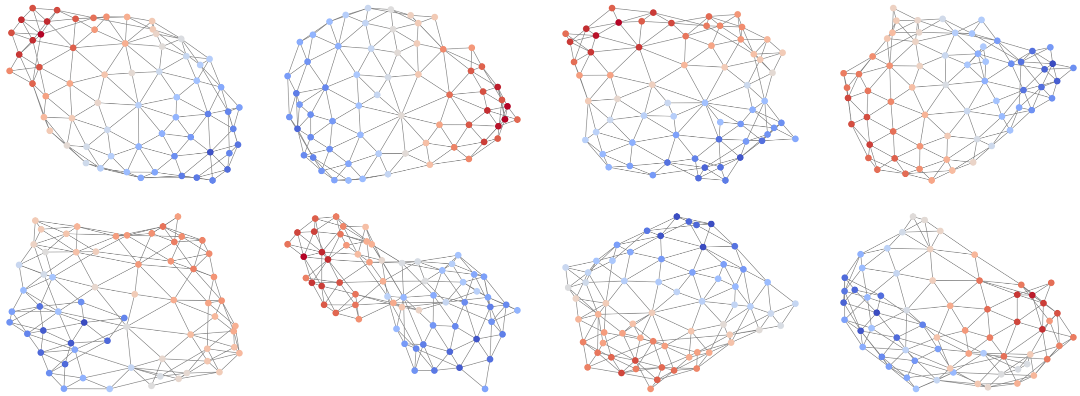


(a) Training graphs.

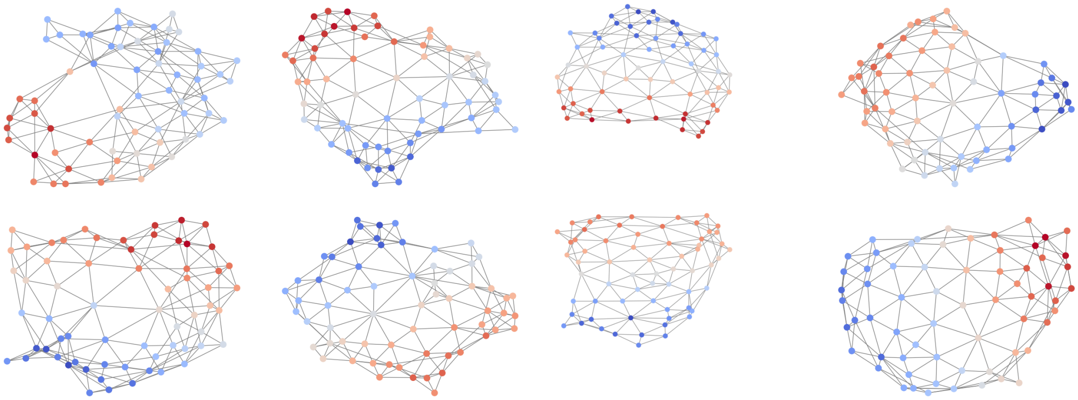


(b) Generated graphs.

Figure 9: Visualization for Moses dataset.

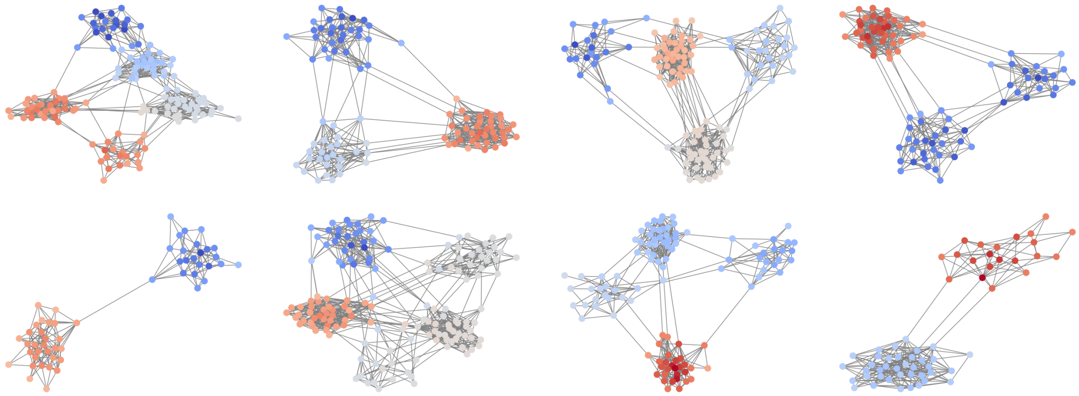


(a) Training graphs.

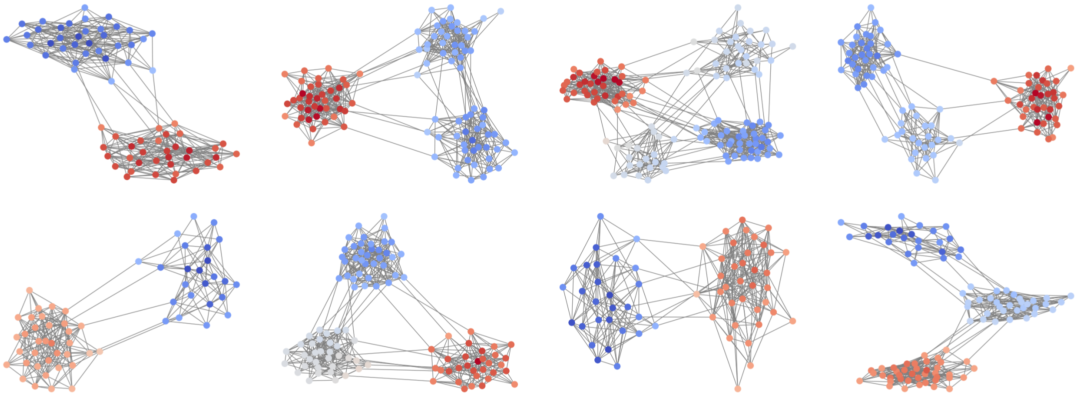


(b) Generated graphs.

Figure 10: Visualization for Planar dataset.

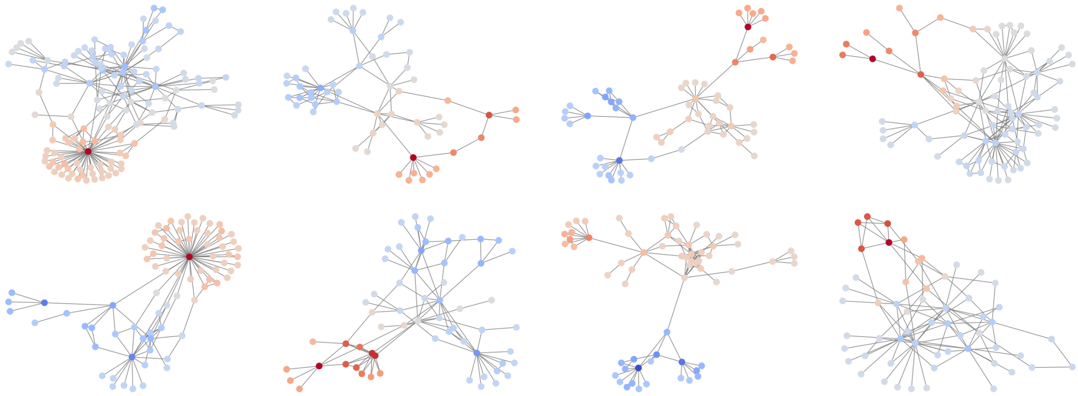


(a) Training graphs.

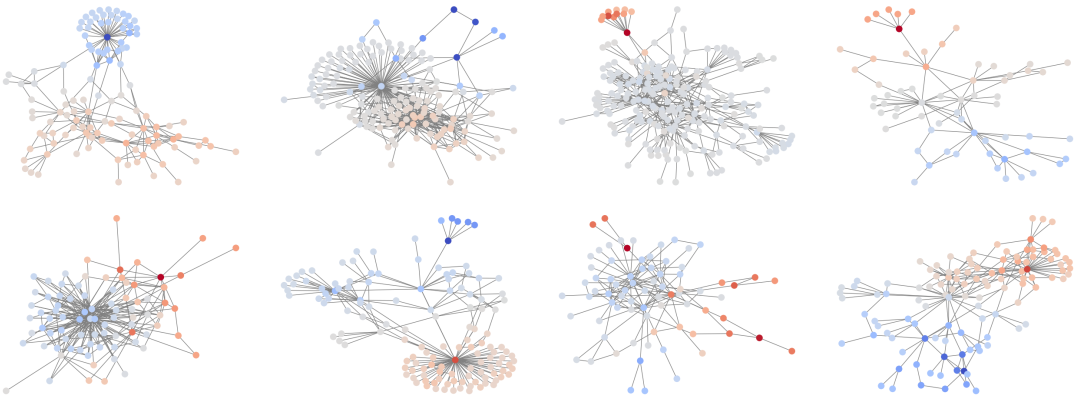


(b) Generated graphs.

Figure 11: Visualization for SBM dataset.

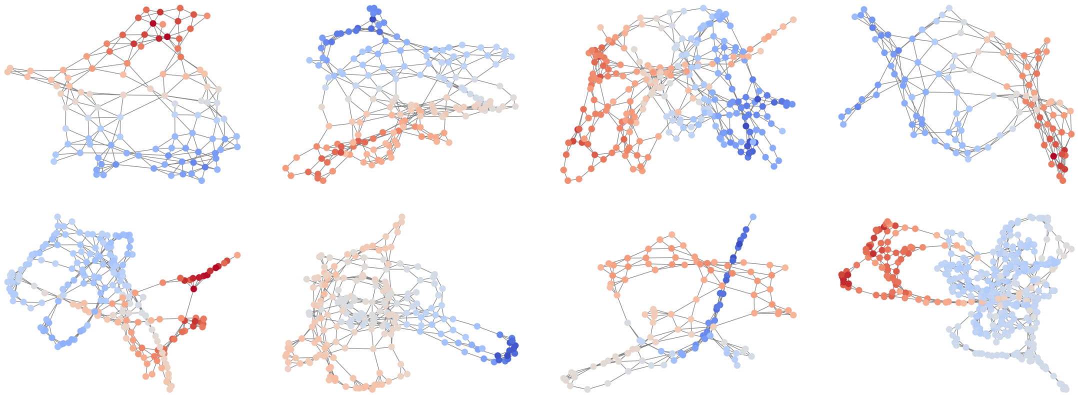


(a) Training graphs.

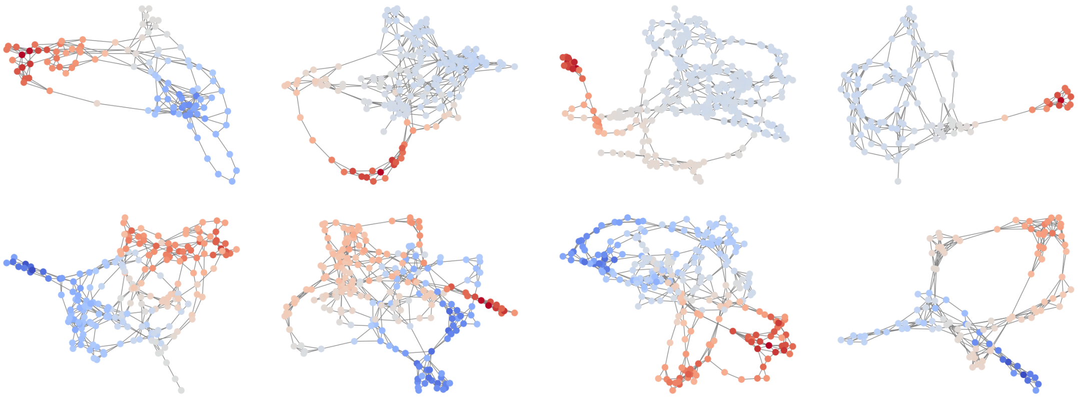


(b) Generated graphs.

Figure 12: Visualization for Ego dataset.



(a) Training graphs.



(b) Generated graphs.

Figure 13: Visualization for Protein dataset.