Editing Partially Observable Networks via Graph Diffusion Models

Puja Trivedi [1] **Ryan A. Rossi** [2] **David Arbour** [2] **Tong Yu** [2] **Franck Dernoncourt** [2] **Sungchul Kim** [2] **Nedim Lipka** [2] **Namyong Park** [3] **Nesreen K. Ahmed** [4] **Danai Koutra** [1]

Abstract

Most real-world networks are noisy and incomplete samples from an unknown target distribution. Refining them by correcting corruptions or inferring unobserved regions typically improves downstream performance. Inspired by the impressive generative capabilities that have been used to correct corruptions in *images*, and the similarities between "in-painting" and filling in missing nodes and edges conditioned on the observed graph, we propose a novel graph generative framework, `SGDM`, which is based on subgraph diffusion. Our framework not only improves the scalability and fidelity of graph diffusion models, but also leverages the reverse process to perform novel, conditional generation tasks. In particular, through extensive empirical analysis and a set of novel metrics, we demonstrate that our proposed model effectively supports the following refinement tasks for partially observable networks: `(T1)` denoising extraneous subgraphs, `(T2)` expanding existing subgraphs and `(T3)` performing "style" transfer by regenerating a particular subgraph to match the characteristics of a different node or subgraph.

1. Introduction

Real-world networks are typically partially observed and noisy (Eliassi-Rad et al., 2019; Kim & Leskovec, 2011; Hanneke & Xing, 2009), posing challenges in downstream tasks (e.g., node classification). Refining a corrupted graph can significantly improve downstream performance but includes a variety of editing tasks with different requirements such as: denoising (removing extraneous edges), expansion (recovering missing edges), and attribute editing (correcting erroneous or missing node attributes). Since the observed graph can be corrupted in different ways, it is important to jointly support all these editing tasks, but existing methods usually only support some of them. For instance, link prediction methods (Zhang & Chen, 2018; Zhang et al., 2020) are able to recover missing edges, but cannot remove noisy edges. Likewise, graph generative adversarial networks (GANs) (Cao & Kipf, 2018; Liao et al., 2019; Wang et al., 2018; Martinkus et al., 2022) can be used to generate candidate subgraphs that maximize a given property by using RL (Cao & Kipf, 2018; Zhu et al., 2022), but it can be difficult to enforce that certain desired subgraphs are preserved in the generated graph (i.e., conditional generation).

Recently, in computer vision, large-scale diffusion models (DMs) (Ho et al., 2020; Hoogeboom et al., 2023; Sohl-Dickstein et al., 2015; Ho et al., 2022) have revolutionized image generation and correction of corrupted images due to (1) their ability to generate extremely high-quality samples while capturing many modes of the underlying data distribution, (2) their ability to easily support powerful conditional generation and editing tasks, such as in-painting, colorization, panorama generation and artifact removal (Lugmayr et al., 2022; Saharia et al., 2022a; Zhang et al., 2023; Ramesh et al., 2021; Saharia et al., 2022b; Kawar et al., 2022; Nie et al., 2022), and (3) their relative ease of training (Dhariwal & Nichol, 2021). Given their success in generating images, there has been significant recent effort in developing graph denoising diffusion models (GDDMs) that accommodate discrete, structured graph data (Austin et al., 2021; Vignac et al., 2023; Chen et al., 2023; Jo et al., 2022), but there is limited work on supporting or even defining analogous graph-based editing tasks.

Inspired by the conceptual similarities between "in-painting" (i.e., filling in masked regions of an image) and filling in missing nodes and edges conditioned on the observed graph, we observe that the distributional modeling capacity of a trained GDDM could be used to refine parts of an observed training graph that may be corrupted or incomplete. However, refining a *single* corrupted network raises additional challenges that preclude the applicability of existing GDDMs; these models assume that they are trained on a *large set of clean graphs*, and have quadratic complexity over the number of nodes preventing them from scaling to more than a few hundred nodes (Jo et al., 2022; Vignac et al.,

[1]CSE Dept, University of Michigan, Ann Arbor [2]Adobe Research Inc [3]Carnegie Mellon University [4]Intel AI Research. Correspondence to: Puja Trivedi <pujat@umich.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

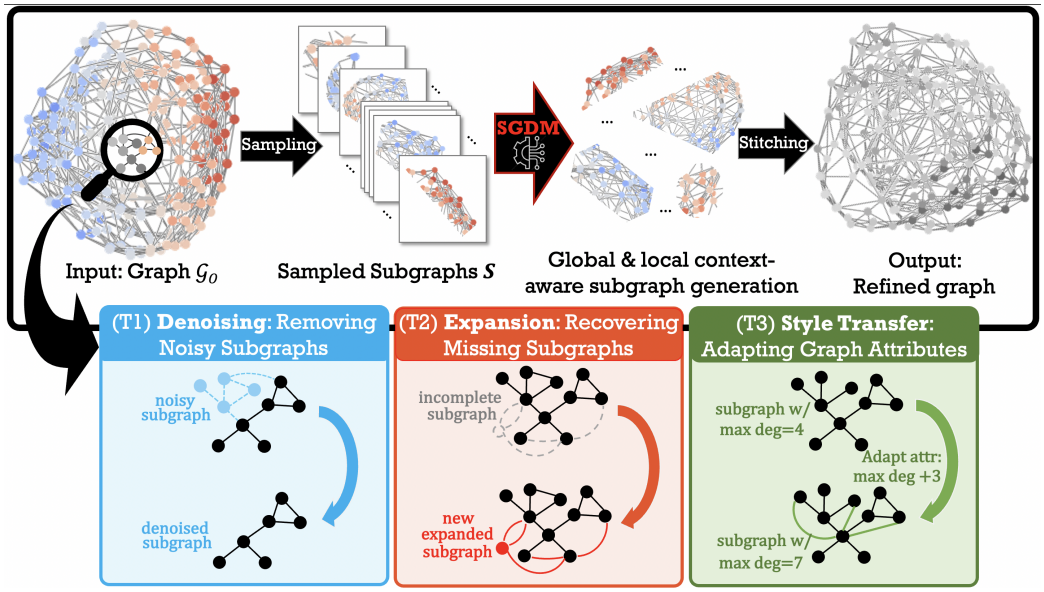

Figure 1. **Overview of Subgraph-Diffusion Models and Editing Tasks.** **SGDM** uses subgraph sampling to flexibly support three useful editing tasks for refining a single, partially observable network.

2023).

Our Work. We focus on the challenging setting of training a GDDM on a single corrupted graph such that the trained model can then be used to perform various conditional graph editing tasks to refine the observed network. In particular, we propose a novel subgraph-based diffusion framework (**SGDM**) that uses graph sampling, and global vs. local contextual information to support three editing tasks: (T1) denoising, (T2) expansion and (T3) structured style-transfer tasks (Fig. 1), which respectively remove extraneous edges, recover missing edges and regenerate graphs to support particular attributes. Our contributions are summarized as:

- **Single Graph Editing Tasks:** We formally introduce three new graph editing tasks—denoising, expansion, style transfer—for single graph GDDMs (§ 3).
- **Subgraph Diffusion Framework:** We propose a novel, diffusion framework that utilizes graph sampling and global vs. local positional information to learn distributions over a single network and support the introduced editing tasks (§ 4).
- **Extensive Empirical Evaluation:** We introduce a diverse set of graph-specific metrics evaluating performance of the editing tasks, and extensively evaluate the ability of GDDMs and our proposed diffusion framework to perform them (§ 5).

2. Related Work & Background

For both image and graph generation tasks, diffusion models (DMs) have outperformed GAN (Cao & Kipf, 2018; Wang et al., 2018; Dhariwal & Nichol, 2021; Liao et al., 2019) and autoencoding alternatives (Simonovsky & Komodakis, 2018; Zahirnia et al., 2022; You et al., 2018) in generating high fidelity, diverse inputs. Conceptually, DMs consist of two parts: (1) a *forward* process which progressively adds noise to a sample and (2) a *reverse* process which learns to remove noise from the corrupted sample. While several improvements have been suggested to *vision* DMs to improve their trainability (Karras et al., 2022; Xiao et al., 2022), fidelity and sampling speed (Rombach et al., 2022; Zhang & Chen, 2023), we provide a generic exposition that also applies to GDDMs (Xie et al., 2021; Mahdi Karami, 2024; Bergmeister et al., 2024) here.

Let $x^0 \sim q(x)$ be a sample from a data distribution. Then, we can define a forward process that gradually adds noise to x^0 over T steps such that in limit x^T can be seen as a sample from a predefined *reference* distribution. Formally, let $q(x^t|x^{t-1})$ be one step of the forward process, then using the Markov property, we can condition on the data sample such that $q(x^1, \ldots, x^T|x^0) = \prod_{t=1}^{T} q(x^t|x^{t-1})$. Image DMs (Sohl-Dickstein et al., 2015) and early graph DMs (Niu et al., 2020; Jo et al., 2022) use an isotropic Gaussian for the reference distribution as it ensures that each step of the reverse process can be computed in closed form, sampling x^T does not require knowledge of x^0, and the posterior has a closed form.

Then, the parameterized reverse process, p_θ, learns to approximate $q(x^{t-1}|x^t)$ to iteratively convert a sample from reference distribution to a sample from $q(x)$. Since each step in the forward process is conditionally independent, we can compute the joint probability over the noisy latent $(x^0, \ldots, x^T)$ as $p_\theta\left(x^{0:T}\right) = p\left(x^T\right) \prod_{t=1}^{T} p_\theta\left(x^{t-1}|x^t\right)$, where marginalizing out the trajectory models the data distribution, $p_\theta(x^0)$. Thus, data can be generated by sampling from the reference distribution, and then using the learned reverse process at each step $T \to 0$. Several equivalent formulations have been proposed for training p_θ by minimizing the variational lower bound, which practically involve training a neural network to predict the noise at step t (Ho et al., 2020; Song & Ermon, 2019; Song et al., 2019). We refer the interested reader to a comprehensive survey by (Yang et al., 2022).

Since graphs are structured, discrete data, alternative reference distributions and corresponding forward processes have been proposed to preserve sparsity and discreteness. For example, DiGRESS (Vignac et al., 2023) proposes a categorical distribution over node and edge types, while EDGE (Chen et al., 2023) uses an empty Erdos-Renyi graph. Both are more scalable than the Gaussian reference distribution used by GDSS (Jo et al., 2022), and produce structured intermediate latent variables. We give more details for graph diffusion models in Appendix A.6.

3. Problem Formulation

In this section, we introduce the editing tasks that our framework supports (Fig. 1).

Notations. Let $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$ represent a graph where the tuple contains, respectively, the node set $\mathcal{V}$, edge set $\mathcal{E}$, adjacency matrix $\mathbf{A} \in [0,1]^{|\mathcal{V}| \; times |\mathcal{V}|}$, and node features $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times d}$.

Problem Statement. Let $\mathcal{G}_O(\mathcal{V}_O, \mathcal{E}_O)$ be an incomplete, noisy *observed* network and $\mathcal{G}_T(\mathcal{V}_T, \mathcal{E}_T)$ be the corresponding, but unknown, complete, noise-free *target* graph. Further, we assume that the majority of the observed graph is not corrupted or missing—i.e., $|(\mathcal{E}_O \setminus \mathcal{E}_T) \cup (\mathcal{E}_T \setminus \mathcal{E}_O)| \leq \min(\epsilon|\mathcal{E}_T|, \epsilon|\mathcal{E}_O|)$, where $\epsilon \ll 1$. Then, given $\mathcal{G}_O$, we seek to devise a scalable, unified GDDM framework that can support `(T1)` expansion, `(T2)` denoising, and `(T3)` style transfer in order to recover the missing subgraphs and remove the extra subgraphs such that the refined graph better aligns with the unknown target graph $\mathcal{G}_T$. While these discrepancies could co-occur, we begin by considering them individually and assume all nodes are shared between the target and observed graphs.

`(T1)` Expansion: Here, we assume that the observed graph is *missing* some subgraph *present* in the target graph. The task is to recover these missing edges given the observed graph. Formally, let $\mathcal{E}_M = \mathcal{E}_T \setminus \mathcal{E}_O$ correspond to the set of missing edges, where $0 < |\mathcal{E}_M| < \epsilon$ and $|\mathcal{V}_O| = |\mathcal{V}_T|$. Then, conditioning on $\mathcal{G}_O$, we wish to generate a graph $\mathcal{G}_G$ that contains both $\mathcal{E}_O$ and $\mathcal{E}_M$.

`(T2)` Denoising: Here, we assume that the observed graph has *extra* edges present that are *not* present in target graph. The task is to remove the additional edges given the observed graph. Formally, let $\mathcal{E}_A = \mathcal{E}_O \setminus \mathcal{E}_T$ correspond to the set of additional edges, where $0 < \mathcal{E}_A < \epsilon$ and $|\mathcal{V}_O| = |\mathcal{V}_T|$. Then, conditioning on $\mathcal{G}_O$, we wish to generate a graph $\mathcal{G}_G$ that removes $\mathcal{E}_A$, but keeps the rest of the edges.

`(T3)` Style Transfer: While expansion and denoising focus on editing $\mathcal{G}_O$ so that the generated graph better aligns with $\mathcal{G}_T$, style transfer is not strictly a refinement task. Instead, it focuses on adapting $\mathcal{G}_O$ to reflect a particular attribute (e.g. style) of interest. We include it in our editing framework since it provides a mechanism for making abstract conditional graph changes. For example, we may be interested in how changing the number of triangles affects connectivity, but do not know apriori which particular edges are changed in response. Formally, let `Attr` be some attribute of interest and $\texttt{Attr}_D$ be the desired value. Then, given $\mathcal{G}_O(\mathcal{V}_O, \mathcal{E}_O)$ with observed value $\texttt{Attr}_O$, we wish to generate a graph, $\mathcal{G}_G(\mathcal{V}_G, \mathcal{E}_G)$, such that $\mathcal{G}_G$ obtains value $\texttt{Attr}_D$. We note that existing GDDMs cannot support this task.

4. Editing Networks with Subgraph Diffusion Models

Here, we describe our subgraph-based diffusion model (**`SGDM`**) and editing framework (Sec. 4.2).

4.1. `SGDM` Framework

Our diffusion framework is motivated by the challenges that existing diffusion models face when training in our problem setup (Sec. 3). Specifically, the majority of image (Saharia et al., 2022a; Ramesh et al., 2021; Saharia et al., 2022b) and graph diffusion models (Vignac et al., 2023; Chen et al., 2023) are trained on very large datasets of generally clean samples to support diverse, high fidelity generation and conditional editing. However, when training diffusion models are applied on a *single* sample (Kulikov et al., 2023; Nikankin et al., 2023), it is easier to overfit to the observed graph—i.e., the reverse process becomes effectively deterministic and only generates the observed graph. This is especially damaging as the observed graph is assumed to be corrupted or otherwise incomplete, and overfitting will lead the diffusion model to memorize these artifacts. Furthermore, while most single-*image* diffusion models assume standard to moderately-sized training images, single-*graph* diffusion models will likely need to accommodate very large training graphs. Lastly, since most parts of the graph are

Algorithm 1 **SGDM**: Subgraph-based Diffusion

Input: $\mathcal{G}_O = (\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$, `Global-Context`, `Local-Context`, `Sampling`, q, p_θ
Output: $(q, p_\theta, \texttt{Hist-Global})$: trained subgraph diffusion model

$\mathbf{C} \leftarrow \texttt{Global-Context}(\mathcal{G}_O)$
$\mathcal{G}_O \leftarrow \mathbf{C}$
$\boldsymbol{\mathcal{S}} \leftarrow \texttt{Sampling}(\mathcal{G}_O)$
$\texttt{Hist-Global} = [0]^{|\mathcal{G}_O|}$
$\texttt{Hist-GraphSize} = [0]^{|\text{Max Size}(|\boldsymbol{\mathcal{S}}|)}$
for $\mathcal{S}$ **to** $\boldsymbol{\mathcal{S}}$ **do**
  $\mathbf{Q} \leftarrow \texttt{Local-Context}(\mathcal{S})$
  $\mathcal{S} \leftarrow \mathbf{Q}$
  $\texttt{Hist-Global}[\mathbf{C}_\mathcal{S}] += 1$
  $\texttt{Hist-GraphSize}[|\mathcal{S}|] += 1$
  $\boldsymbol{\mathcal{S}} \leftarrow (\mathcal{S} = \mathbf{A}_\mathcal{S}, \mathbf{X}_\mathcal{S}, \mathbf{C}_\mathcal{S}, \mathbf{Q})$
end
$\texttt{Forward} \leftarrow q\left(\mathcal{S}^1, \ldots, \mathcal{S}^T\right) = \prod_{t=1}^T q\left(\mathcal{S}^t | \mathcal{S}^{t-1}\right)$
$\texttt{Reverse} \leftarrow p_\theta\left(\mathcal{S}^{0:T}\right) = p(\mathcal{S}^T) \prod_{t=1}^T p_\theta\left(\mathcal{S}^{t-1} | \mathcal{S}^t, \mathbf{C}_{\mathcal{S}^0}\right)$
$\min_\theta \sum_{\mathcal{S} \in \boldsymbol{\mathcal{S}}} \mathbb{E}_{q(\mathcal{S}_0) q(\mathcal{S}_{1:T} | \mathcal{S}_0)} \left[-\log \frac{p_\theta(\mathcal{S}_{0:T})}{q(\mathcal{S}_{1:T} | \mathcal{S}_0)} \right]$
return $(q, p_\theta, \texttt{Hist-Global}, \texttt{Hist-GraphSize})$

Algorithm 2 Editing with **SGDM**

Input: $\mathcal{S}_O = (\mathcal{V}_O, \mathcal{E}_O, \mathbf{A}_O, \mathbf{X}_O, \mathbf{C}_O)$: subgraph that will be edited, $f_\psi : (\mathcal{S}, t) \to y$: trained attribute regressor, λ: guidance strength
Output: $\mathcal{S}_G = (\mathbf{A}_G, \mathbf{X}_G, \mathbf{C}_O)$: edited graph.

$n \leftarrow |\mathcal{S}_O|$
$\mathcal{S}^{t=T} \leftarrow q_n(\cdot)$
for $t = T$ **to** 0 **do**
  $\mathbf{Q}^t \leftarrow \texttt{Local-Context}(\mathcal{S}^t)$
  $\mathcal{S}^t \leftarrow \mathbf{Q}^t$
  $\mathcal{S}^{t-1} \leftarrow p_\theta(\mathcal{S}^t, \mathbf{C})$
  $(\mathbf{A}^{t-1}, \mathbf{X}^{t-1}, \mathbf{C}) \leftarrow \mathcal{S}^{t-1}$
  ▷ `Expansion.`
  $\mathbf{A}^{t-1} \leftarrow (1 - \mathbf{A}) \odot \mathbf{A}^{t-1} + \mathbf{A}$
  $\mathbf{X}^{t-1} \leftarrow (1 - \mathbf{X}) \odot \mathbf{X}^{t-1} + \mathbf{X}$
  ▷ `Denoising.`
  $\mathbf{A}^{t-1} \leftarrow \mathbf{A} \odot \mathbf{A}^{t-1}$
  ▷ `Style Transfer. (Vignac et al., 2023; Dhariwal & Nichol, 2021)`
  $\widetilde{y} = f(\mathcal{S}^t, t)$
  $p_\psi(\widetilde{y} | \mathcal{S}^{t-1}) \propto exp(-\lambda \langle \nabla \mathcal{S}^t || \widetilde{y} - y||^2, \mathcal{S}^{t-1} \rangle)$
  $\mathcal{S}^{t-1} \sim p_\theta(\mathcal{S}^{t-1} | \mathcal{S}^t) p_\psi(\widetilde{y} | \mathcal{S}^{t-1})$
end
return $\mathcal{S}_G = (\mathbf{A}^t, \mathbf{X}^t, \mathbf{C}_O)$

assumed to be complete and uncorrupted, it is unnecessary to regenerate the entire graph per edit.

We propose Subgraph-based Diffusion Models (**SGDM**) which rely upon *subgraph sampling* and *global context* to address these challenges, and flexibly support existing GDDMs in generating and editing large-scale networks (Algorithms 1). Next, we discuss the key components of **SGDM**.

Subgraph Sampling. Existing GDDMs have demonstrated impressive performance in modeling large distributions of *small* graphs but are limited by the quadratic memory and runtime complexity incurred by considering every potential edge during generation. This is further exacerbated when using expressive but memory-intensive graph transformer architectures during training. While there has been some recent work on improving the scalability of GDDMs by only taking the gradient with respect to a subset of active nodes (Chen et al., 2023), such approaches are nonetheless destined to run out-of-memory on production-scale networks as even a subset of active nodes will be too large to compute the backward pass. Moreover, as more sophisticated GDDMs are proposed, it is beneficial to support these methods on large-scale networks as well.

To this end, we use subgraph sampling and stitching to convert the task of learning on a single large-scale network to learning a distribution over a collection of subgraphs. Namely, given the observed graph, $\mathcal{G}_O$, we use a subgraph sampling function: $\texttt{SAMPLE} : \mathcal{G}_O \to \boldsymbol{\mathcal{S}}$, to return a set of subgraphs, $\boldsymbol{\mathcal{S}}$, such that (1) the average size of a subgraph $\mathcal{S}$ is much smaller than $\mathcal{G}_O$, i.e., $\left(\frac{1}{|\boldsymbol{\mathcal{S}}|} \sum_{\mathcal{S} \in \boldsymbol{\mathcal{S}}} |\mathcal{S}|\right) \lll |\mathcal{G}_O|$; (2) each node is represented *at least* once through sampling—$\forall v \in \mathcal{V}_O, |\{\mathcal{S} | v \in (\mathcal{S} \ni \boldsymbol{\mathcal{S}}\}| > 1$; and (3) for trainability, $\mathcal{S}$ is connected. Graph partitioning and clustering methods, as well as ego-network sampling are candidates for sampling, potentially followed by post-processing to ensure the connectivity requirement. In case the obtained subgraphs remain prohibitively large, an optional subsampling function can be applied to reduce the size of $\mathcal{S}$ to a predefined maximum size N_{max}. Formally, define $\texttt{SUBSAMPLE} : \mathcal{S} \to \widetilde{\mathcal{S}}$, such that $|\widetilde{\mathcal{S}}| \leq N_{max}$ and $\widetilde{\mathcal{S}}$ remains connected.

Given the subgraphs, we can use any existing GDDM to define p_θ, q, and the reference distribution (Vignac et al., 2023; Chen et al., 2023; Jo et al., 2022) over $\mathcal{S}$ as follows, including those that may have been previously prohibited through the large memory complexity:

$$\text{Forward Process: } q\left(\mathcal{S}_k^t(v) | \mathcal{S}_k^{t-1}(v)\right) \tag{1}$$

$$\text{Reverse Process: } p_\theta\left(\mathcal{S}_k^{t-1}(v) | \mathcal{S}_k^t(v)\right) \tag{2}$$

$$\text{Joint Prob: } p\left(\mathcal{S}_k^T(v)\right) \prod_{t=1}^T p_\theta\left(\mathcal{S}_k^{t-1}(v) | \mathcal{S}_k^t(v)\right). \tag{3}$$

We note that while our primary objective is *not* to learn a large-network generative model, **SGDM** can be used for this purpose as it has been shown that it is possible to reconstruct a graph from a reasonably-sized collection of its random subgraphs with sufficient coverage (Manvel, 1976; McGregor & Sengupta, 2022). We discuss "stitching" at the end of the subsection.

Context. While Equations (1,2,3) define a complete graph diffusion model over $\boldsymbol{\mathcal{S}}$, we can further improve its generative capabilities by introducing information that can differentiate between similar subgraphs. It is possible that there exist subgraphs $\mathcal{S}_i, \mathcal{S}_j \in \boldsymbol{\mathcal{S}}$ which are structurally similar but located in different parts of the network. To

handle such cases effectively, we design **SGDM** to leverage both *global* context (e.g., how close $\mathcal{S}_i$ is to $\mathcal{S}_j$ in $\mathcal{G}_O$, what $\mathcal{S}_i$'s surroundings are) and *local* context (i.e., the structural characteristics of $\mathcal{S}_i$) for editing tasks. For example, when editing a corrupted subgraph, it may be more helpful to leverage information from a closer, structurally similar subgraph than one that is further away; global context helps us capture this and ensures the edited graph remains consistent with respect to its surroundings.

To improve model performance and effectively provide `Local-Context` : $\mathcal{S} \rightarrow \mathbb{R}^{|\mathcal{S}| \times d}$, existing GDDMs (Vignac et al., 2023; Chen et al., 2023) use spectral and structural features (e.g., cycle counts, number of triangles, Laplacian vectors, degree distributions). For global context, let `Global-Context` : $\mathcal{G}_O \rightarrow \mathbb{R}^{|\mathcal{G}_O| \times d}$ be a function that computes a global-context vector ($\mathbf{C}_v$) for each node $v \in \mathcal{V}_O$ such that (i) $\forall u, v, q, u \neq v \in \mathcal{V}_O, \mathbf{C}_u \neq \mathbf{C}_v$, and (ii) for most $u, v, (u \neq v) \in \mathcal{V}_O$, $|\mathbf{C}_u$ - $\mathbf{C}_v| \leq |\mathbf{C}_v$ - $\mathbf{C}_q|$ if $d(u, v) \leq d(v, q)$ for some graph distance d. To incorporate global information, we modify p_θ to take $\mathbf{C}_\mathcal{S}$ as an additional feature (as described in the appendix). Notably, when generating new samples, we first sample n subgraph global-context vectors from the empirical distribution of global-context vectors in $\boldsymbol{\mathcal{S}}$, and then proceed with the reverse process. This allows us to control the broader context in which graphs are generated. (Algorithm 1).

Stitching. As we mentioned above, it is possible to reconstruct graphs given a collection of random subgraphs (Manvel, 1976; Gupta & Khandelwal, 2012; McGregor & Sengupta, 2022) with enough samples and provided there is a mechanism for identifying nodes across the subgraphs so that edges may be coalesced. Since we required `Global-Context` to be unique $\forall v \in \mathcal{V}_O$, we can use the global-context vectors to perform this identification and then define a `COALESCE`: $\widetilde{\boldsymbol{\mathcal{S}}} \rightarrow \widetilde{\mathcal{G}_O}$ function accordingly, e.g., take the union over collection of edges. (Algorithm 3).

4.2. Supported Editing Tasks

We now discuss how to perform the editing tasks introduced in Sec. 3 for both **SGDM** and single-graph diffusion models. We assume that we have a pretrained diffusion model, p_θ, and the portion of the graph that needs to be edited is provided, $\mathcal{S}(\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X}, \mathbf{C})$.

`(T1)` **Expansion with SGDM.** For expansion, the reverse process needs to recover missing edges or subgraphs, given the specified, observed subgraph. Conceptually, this task is similar to "in-painting" where a diffusion model is used to fill-in masked regions of an image so that the resulting image is consistent with the surrounding context. Our approach is inspired by a computer vision approach (Lugmayr et al., 2022) that uses the DM to only fill in the masked regions of the images; it only modifies the reverse process and ensures consistency with the starting sample by incorporating information from the unmasked regions throughout the reverse process. For graphs, the masked region is defined by the connectivity and node features of the provided, starting subgraph, $\mathcal{S}(\mathbf{A}, \mathbf{X}, \mathbf{Q})$, and unmasked region is implicitly defined by the potential $|\mathcal{S}|^2$ - $\mathbf{A}$ edges of $\mathcal{S}$. Formally, for each step t in the reverse process, $\mathbf{A}^{t-1} \leftarrow (1 - \mathbf{A}) \odot \mathbf{A}^{t-1} + \mathbf{A}$, where the full process is shown in Algorithm 2.

`(T2)` **Denoising with SGDM.** While expansion assumes that all the existing edges $\mathcal{S}(\mathbf{A}, \mathbf{X}, \mathbf{C})$ are reliable, and can be used to guide the reverse process, this is not the case when performing denoising. Indeed, when performing denoising, it is unknown *which* subset of $\mathbf{A}$ can be considered reliable. This makes it challenging to ensure that the generated sample is consistent with the starting sample. On the other hand, while expansion requires correctly identifying which of the possible $|\mathcal{S}|^2$ - $\mathbf{A}$ edges should be added to $\mathcal{S}$, denoising requires only identifying which of the $\mathbf{A}$ edges are in fact correct. Since graphs are often sparse, this entails that the possible "solution" space when denoising is significantly smaller than that when performing expansion. We propose to use an in-painting style approach, but instead of filling-in regions given $\mathcal{S}$, we attempt to remove edges that do not belong: $\mathbf{A}^{t-1} \leftarrow \mathbf{A} \odot \mathbf{A}^{t-1}$. This ensures that the sample generated at the end of the reverse process will only contain edges that belong to $\mathcal{S}$, and, due to the impressive generative capacity of the pretrained diffusion model, resulting sample may also remove the extraneous edges.

`(T3)` **Style Transfer with SGDM.** Consider an example where we are interested in understanding how changing a node's attribute will lead to changes in its connectivity. With an existing single large-graph GDDM, the entire graph would be regenerated in an attempt to reflect these changes despite the expectation that edits will be relatively local (Newman, 2010). In contrast, with **SGDM**, we can provide the relevant subgraph (i.e., k-hop egonet) of the specified node and regenerate to reflect the changed attribute. This is particularly important for scalability as we do not consider an in-painting approach here, but instead leverage classifier guidance (Dhariwal & Nichol, 2021; Kawar et al., 2023; Vignac et al., 2023) which requires taking the gradient of the generated graph while attempting to minimize the loss of a time-dependent classifier. We build on a conditional generation pipeline (Vignac et al., 2023) that is unable to perform style transfer on large networks due to memory issues during training. To overcome these challenges, in brief, we first train a time-dependent regressor, $f_\psi(\mathcal{S}, t)$ to predict the attribute of interest, then we use the regressor loss, $(p_\psi(\widetilde{y}|\mathcal{S}^{t-1}))$, to guide the reverse process $\mathcal{S}^{t-1} \sim p_\theta(\mathcal{S}^{t-1}|\mathcal{S}^t)p_\psi(\widetilde{y}|\mathcal{S}^{t-1})$ (Algorithm 2).

Table 1. **Dataset Statistics**

Dataset	# Graphs	# Nodes	# Edges
BA-Shapes	1	700	3,944
PolBlogs	1	1,222	33,431
CORA	1	2,485	10,138

5. Experiments

In this section, we evaluate the ability of graph diffusion models to perform editing tasks, and demonstrate the benefits of using **SGDM**. Specifically, we seek to answer the following research questions: **(RQ1)** Is there a benefit to SAMPLING on the expansion and denoising tasks? **(RQ2)** Is there a benefit to using GLOBAL-CONTEXT in our proposed graph editing tasks? **(RQ3)** How do different GDDM backbones affect the performance of **SGDM**? Below, we first describe our evaluation setup.

5.1. Evaluating Refinement Tasks

To the best our knowledge, we are the first to consider using graph editing to perform refinement on a single, partially observed network. Therefore, we begin by introducing the following new metrics.

5.1.1. Novel Evaluation Metrics

Evaluating the ability of generative graph models to recover $\mathcal{G}_T(\mathcal{V}_T, \mathcal{E}_T)$ through conditional editing tasks presents several unique difficulties in comparison to similar editing tasks in computer vision. While repainted images are assessed for "realism", which considers how plausible an in-painting solution is given the unmasked region, and "diversity", which considers the perceptual dissimilarity of generated solutions, it is non-trivial to define such quantities for edited graphs. Indeed, in the case of a subgraph missing a single edge, generating a diverse set of alternative subgraphs may introduce unnecessary changes that do not align with $\mathcal{G}_T$. Furthermore, it is not possible to perform user-studies or visual inspection to determine if the edited graph maps to realistic solutions due to their size and the abstract nature of structured data. To this end, we introduce the following novel metrics for assessing the quality of expanded or denoised graphs. We discuss style transfer evaluation separately.

• **Consensus:** If the score function has learned to model the true data-generating process underlying the unobserved target graph, then it is expected that the majority of generated samples will concur that the missing edges should be generated. A higher consensus score, $\frac{1}{|\mathcal{E}_M|}\sum_{[i,j]\in\mathcal{E}_M}\frac{1}{R}\sum_{r=1}^{R}\mathbb{1}\left[{\mathcal{E}_G}^r_{[i,j]} = \mathcal{E}_{T[i,j]}\right]$, where R is the number of generated samples, indicates that the majority of samples are able to generate the particular set of missing edges, as desired.

• **Edge Overlap:** While **SGDM** allows us to generate aligned graphs by conditioning with the positional information, we note that other methods may not produce such graphs. Therefore, after performing some alignment, we measure how much of the generated graph overlaps with the conditioning graph. Since measuring overlap is an NP-hard problem, we approximate it as $\frac{1}{R}\sum_{r=1}^{R}\mathbb{1}\left[{\mathcal{E}_G}^r \odot \mathcal{E}_O\right]$, using the binary adjacency matrix representation of the graph.

• **Diversity:** While producing a diverse set of generated solutions may not be as high priority in graph editing tasks as in image editing tasks, it is still useful to measure to understand if the generative model has converged to a single solution or collapsed to single model: $\frac{1}{R(R-1)}\sum_{r_1=1}^{R}\sum_{r_2=1}^{R}\mathbb{1}\left[{\mathcal{E}_G}^{r_1} \neq {\mathcal{E}_G}^{r_2}\right]$.

• **Sparsity:** Given that we seek to recover missing edges conditioned on the observed graph, it is possible to trivially recover all missing edges if the generated samples are fully connected. Thus, we measure the fraction of extra edges generated with respect to the target graph: $\frac{1}{R}\sum_{r=1}^{R}\frac{|\mathcal{E}_G^r|}{|\mathcal{E}_T|}$.

While the desired sparsity score is closer to 1, the range of sparsity is $[\frac{|\mathcal{E}_O|}{|\mathcal{E}_T|}, \frac{|\mathcal{V}_O|^2}{|\mathcal{E}_T|}]$, corresponding to generated samples which do not add any edges given the conditioning graph and those that generate all possible edges.

5.1.2. Experimental Setup

Datasets. For the editing tasks, we consider 3 large, single networks—BA-Shapes, PolBlogs and CORA (Table 1)—and corrupt them to create the incomplete, noisy observed graphs. Namely, we randomly add or remove 10% of edges in the CORA or Pol-Blogs dataset, while enforcing that the graph remains connected. In order to assess performance on a structured corruption process, we also evaluate the BA-Shapes dataset (Ying et al., 2019), which consists of 80 house-motifs attached to a 300-node Barbasi Albert random graph. Here, we corrupt 5% of the edges in the house-motifs. Since the house-motifs have a fixed structure, we can better understand if the diffusion model is able to capture the underlying data-generating process in order to refine the graph. Given the set of the removed or added edges, we are able to compute the metrics introduced above (Sec. 5.1).

Baselines. Given that editing a single network is a relatively new problem, there are no direct baselines, so we adapt existing graph diffusion models to the best of our abilities. EDGE (Chen et al., 2023), which conditions the reverse process using the empirical degree distribution is our most direct comparison and, as we discuss below, helps us show the benefits of **SGDM**.

SGDM Variants. While **SGDM** was presented as a general framework in Sec. 4.1, we evaluate specific variants that are well-suited for our editing tasks. We define SAMPLING using node-centric 2-hop ego-networks; SUBSAMPLING by (1) randomly sampling 200 nodes per subgraph, (2) inducing the corresponding subgraph, and (3) taking the largest connected component; and GLOBAL-Context by computing the top-two Laplacian positional encodings (LPEs) (Dwivedi et al., 2020; 2022; Wang et al., 2022) on $\mathcal{G}_O$. We select ego-networks for SAMPLING as it aligns well with use cases where particular nodes are untrustworthy or corrupted. Our SUBSAMPLING strategy is selected for ease of use and efficiency. We chose LPEs for global context as they have been shown to be theoretically expressive in distinguishing between nodes. We consider 3 different backbone GDDMs (DiGRESS (Vignac et al., 2023), EDGE (Chen et al., 2023) and GDSS (Jo et al., 2022)) in our evaluation to demonstrate the flexibility of the proposed framework. Local-Context is defined according to backbone GDDM. We discuss more variants in Appendix A.8.

5.1.3. Results

Here, we answer the aforementioned research questions. Scores closer to "1" indicate better performance.

(RQ1) Benefits of Sampling. In Table 2, EDGE is compared against **SGDM** when using a EDGE backbone and NO GLOBAL-CONTEXT to determine if learning over a collection of subgraphs is more effective than directly learning on a single graph. We observe that **SGDM** significantly outperforms EDGE on Diversity, Sparsity and Edge Overlap on the denoising task. While EDGE achieves better Consensus, it does so by removing too many edges. On expansion, both struggle to perform well. We posit that this is a side-effect of guiding the reverse process using a degree distribution, as other GDDM backbones perform better.

(RQ2) Benefits of Global-Context. In Table 3, we see that **SGDM** (with GLOBAL-CONTEXT) continues to outperform EDGE with or without GLOBAL-CONTEXT. Moreover, compared against **SGDM** (without GLOBAL-CONTEXT) in Table 2, we see that including context improves Sparsity and Edge-Overlap on both Cora and PolBlogs. Recall that context also enables effective stitching for large network generation, though it is not the main focus of this work.

(RQ3) Effect of Backbone. To demonstrate the flexibility of our proposed framework, we evaluate the performance of **SGDM** on expansion when using GDSS and DiGRESS backbones (see A.9 for results on denoising). In Fig. 2, we see that both DiGRESS and GDSS achieve better Consensus and Diversity than EDGE. While GDSS has significantly greater Sparsity than other methods, this is not unexpected as GDSS treats the adjacency matrix as a continuous variable, leading to a densely generated graph. Nonetheless, **SGDM** continues to outperform EDGE, the only comparable single-graph method.

Table 2. **Benefits of SGDM.** Compared against EDGE, we see that using **SGDM** +EDGE (gray rows) improves denoising performance. EDGE uses the degree distribution to guide the reverse process. This has negative effects on (T1) expansion, as it constrains the model to satisfy an expected number of edges. Using other DM backbones helps resolve this problem.

Dataset	Task	SGDM	Global-Ctx	Consensus	Diversity	Sparsity	Edge Overlap
BA-Shapes	Denoising	✗	✗	**0.954**	0.834	0.055	0.0427
BA-Shapes	Denoising	✓	✗	0.751	**0.992**	**0.364**	**0.3427**
BA-Shapes	Expansion	✗	✗	0.000	0.100	0.930	1.000
BA-Shapes	Expansion	✓	✗	**0.055**	0.100	0.930	1.000
Cora	Denoising	✗	✗	**0.916**	0.915	0.115	0.0957
Cora	Denoising	✓	✗	0.587	**1.000**	**0.591**	**0.5851**
Cora	Expansion	✗	✗	0.000	0.100	0.922	1.000
Cora	Expansion	✓	✗	0.000	0.100	0.925	1.000
PolBlogs	Denoising	✗	✗	**0.992**	0.702	0.0411	0.0412
PolBlogs	Denoising	✓	✗	0.713	**1.000**	**0.340**	**0.3156**
PolBlogs	Expansion	✗	✗	0.000	0.100	0.941	1.000
PolBlogs	Expansion	✓	✗	0.000	0.100	0.939	1.000

Table 3. **Benefits of Global-Context.** We see that **SGDM** with Global-Context (gray rows) continues to outperform single-graph EDGE with or without additional context. Recall that context is also useful for stitching to generate large graphs.

Dataset	Task	SGDM	Global-Ctx	Consensus	Diversity	Sparsity	Edge Overlap
BA-Shapes	Denoising	✗	✗	0.954	0.834	0.055	0.0427
BA-Shapes	Denoising	✗	✓	**0.996**	0.733	0.019	0.0152
BA-Shapes	Denoising	✓	✓	0.341	**0.928**	**0.762**	**0.7210**
BA-Shapes	Expansion	✗	✗	0.000	0.100	0.930	1.000
BA-Shapes	Expansion	✗	✓	0.000	0.100	0.930	1.000
BA-Shapes	Expansion	✓	✓	**0.055**	0.100	0.930	1.000
Cora	Denoising	✗	✗	0.916	0.915	0.115	0.0957
Cora	Denoising	✗	✓	**0.965**	0.798	0.057	0.0486
Cora	Denoising	✓	✓	0.754	**1.000**	**0.442**	**0.4376**
Cora	Expansion	✗	✗	0.000	0.100	0.922	1.000
Cora	Expansion	✗	✓	0.000	0.100	0.922	1.000
Cora	Expansion	✓	✓	0.000	0.100	0.918	1.000
PolBlogs	Denoising	✗	✗	**0.992**	0.702	0.0411	0.0412
PolBlogs	Denoising	✗	✓	**0.993**	0.598	0.023	0.0218
PolBlogs	Denoising	✓	✓	0.375	**1.000**	**0.640**	**0.5927**
PolBlogs	Expansion	✗	✗	0.000	0.100	0.941	1.000
PolBlogs	Expansion	✗	✓	0.000	0.100	0.938	1.000
PolBlogs	Expansion	✓	✓	0.000	0.100	0.939	1.000

5.2. Graph Style Transfer Results

Unlike (T1) expansion and (T2) denoising, which seek to use the generative model to refine subgraphs so that they may better align with $\mathcal{G}_T$, (T3) Style Transfer instead modifies subgraphs so that they may align with some specified property of interest. Thus, we introduce a task-specific experimental and evaluation setup.

Style Transfer Tasks. We consider three different graph properties to modify through style-transfer: (1) Sum of Degrees: Given a subgraph with total degree D_T, we wish to modify it so that the resultant graph has $D_T + 3$ as the total degree, (2) Maximum Degree: Given a subgraph with maximum degree D_M, we wish to modify it so that the result graph has $D_M + 3$ as the maximum degree, and (3) Number of Triangles: Given a subgraph with triangle count τ_c, we wish to modify it so that the result graph has $\tau_c + 3$ as

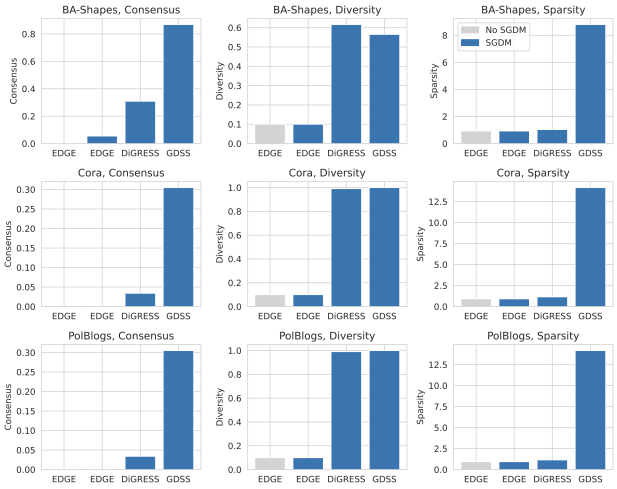

Figure 2. **Choice of Diffusion Models.** **SGDM** continues to outperform EDGE when using GDSS and DiGRESS as backbones. Indeed, these backbones improve the Consensus and Diversity as well.

Table 4. **Graph Style Transfer.** While **SGDM** can support localized style transfer, single-graph DMs like EDGE cannot. Moreover, we find that incorporating `Global-Context` can helps improve performance on two out of three style transfer tasks.

Task	Global-Context?	Regressor Type	MAE(↓)
Sum-Degree	✗	GCN	6.46
	✗	GIN	6.81
	✗	GTrans	6.48
	✓	GCN	3.78
	✓	GIN	3.62
	✓	GTrans	**3.45**
Max-Degree	✗	GCN	24.73
	✗	GIN	45.09
	✗	GTrans	24.63
	✓	GCN	**9.3**
	✓	GIN	11.04
	✓	GTrans	11.44
Num. Triangles	✗	GCN	**9.18**
	✗	GIN	9.57
	✗	GTrans	10.29
	✓	GCN	19.6
	✓	GIN	19.6
	✓	GTrans	19.6

the total number of triangles. These properties are selected because they can correspond to abstract concepts that may be relevant to downstream tasks. For example, in a social network, changing the sum of degrees or maximum degree may simulate a new user becoming more connected over time, while changing the number of triangles can reflect how friends-of-friends may become connected.

Experimental Setup. For all style transfer tasks, we first train a regressor, $p(y|G_t, t)$ that predicts the target (e.g., max degree) from the graph at time t, and use it with $\lambda = 100$ to guide the reverse process as shown in Algorithm 2. Given that predicting graph properties using GNNs can be difficult, we report results for three different regressor backbones—GCN, GIN, and Graph Transformer. Furthermore, we consider generative models that have been trained with and without positional encodings. We emphasize that while our setup does follow that of Vignac et al. and Dhariwal & Nichol, without the SGDM framework this task is meaningless on single networks, as localized edits cannot be performed. **Evaluation.** In Table 4, we report the average of the mean absolute error between the target property of interest and the resulting property from the generated graph for 100 samples. Here we show results on the BAShapes dataset when DiGRESS is used as the underlying diffusion model, and provide additional results including qualitative examples in the A.9.

Results. We make the following observations from Table 4. First, we observe that using `GLOBAL-CONTEXT` improves performance on the Sum-Degree and Max-Degree tasks, irrespective of the choice of regressor architecture. However, `GLOBAL-CONTEXT` is not as helpful for changing the number of triangles. We hypothesize that the LPEs may encode information that makes it difficult to change the number of triangles. Lastly, we note that while the choice of regressor architecture does not significantly impact the performance when `GLOBAL-CONTEXT` is used, it is influential when they are not used, suggesting that incorporating them by help stabilize training.

6. Conclusion

We proposed a novel subgraph-based diffusion model (**SGDM**) to support editing tasks on a single, partially observable network. In particular, **SGDM** uses subgraph sampling and global context to convert single-graph training into learning over a collection of subgraphs and is flexible to choice of diffusion backbone. Notably, this allows us to use expressive GDDM backbones, such as DiGRESS, that were previously prohibited due to quadratic memory requirements. We also formalized three conditional graph editing tasks—namely, expansion, denoising, and style transfer—that can be used for graph refinement with our framework. Lastly, we introduced a new evaluation protocol for our problem setting and performed extensive empirical evaluation to demonstrate the benefits of our proposed framework, as well as the utility of the proposed editing tasks. For future work, we will expand our evaluation to include downstream task performance and our framework to support editing dynamic graphs.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

Acknowledgements

This work was performed while PT was an intern at Adobe Research. DK is also partially supported by the National Science Foundation under CAREER Grant No. IIS 1845491, Army Young Investigator Award No. W9-11NF1810397, and Adobe, Amazon, Facebook, and Google faculty awards. Any opinions, findings, and conclusions or recommendations expressed here are those of the author(s) and do not reflect the views of funding parties. PT thanks Ekdeep Singh Lubana for useful discussions during the course of this project.

References

Austin, J., Johnson, D. D., Ho, J., Tarlow, D., and van den Berg, R. Structured denoising diffusion models in discrete state-spaces. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2021.

Bergmeister, A., Martinkus, K., Perraudin, N., and Wattenhofer, R. Efficient and scalable graph generation through iterative local expansion. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2024.

Cao, N. D. and Kipf, T. Molgan: An implicit generative model for small molecular graphs. *CoRR*, abs/1805.11973, 2018.

Chanpuriya, S., Musco, C., Sotiropoulos, K., and Tsourakakis, C. E. On the power of edge independent graph models. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2021.

Chen, X., He, J., Han, X., and Liu, L. Efficient and degree-guided graph generation via discrete diffusion modeling. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2023.

Dhariwal, P. and Nichol, A. Q. Diffusion models beat gans on image synthesis. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2021.

Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking graph neural networks. *CoRR*, abs/2003.00982, 2020.

Dwivedi, V. P., Luu, A. T., Laurent, T., Bengio, Y., and Bresson, X. Graph neural networks with learnable structural and positional representations. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2022.

Eliassi-Rad, T., Caceres, R. S., and LaRock, T. Incompleteness in networks: Biases, skewed results, and some solutions. In *Proc. Int. Conf. on Knowledge Discovery & Data Mining (KDD)*, 2019.

Fey, M. and Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Gupta, S. K. and Khandelwal, A. Reconstruction conjecture for graphs isomorphic to cube of a tree. *CoRR*, abs/1207.1875, 2012.

Hanneke, S. and Xing, E. P. Network completion and survey sampling. In *Proc. Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2009.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2020.

Ho, J., Saharia, C., Chan, W., Fleet, D. J., Norouzi, M., and Salimans, T. Cascaded diffusion models for high fidelity image generation. *J. Mach. Learn. Res. (JMLR)*, 23, 2022.

Hoogeboom, E., Nielsen, D., Jaini, P., Forré, P., and Welling, M. Argmax flows and multinomial diffusion: Learning categorical distributions. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2021.

Hoogeboom, E., Heek, J., and Salimans, T. simple diffusion: End-to-end diffusion for high resolution images. *arXiv preprint arXiv:2301.11093*, 2023.

Jo, J., Lee, S., and Hwang, S. J. Score-based generative modeling of graphs via the system of stochastic differential equations. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2022.

Jo, J., Kim, D., and Hwang, S. J. Graph generation with destination-driven diffusion mixture. *CoRR*, abs/2302.03596, 2023.

Karras, T., Aittala, M., Aila, T., and Laine, S. Elucidating the design space of diffusion-based generative models. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2022.

Kawar, B., Song, J., Ermon, S., and Elad, M. JPEG artifact correction using denoising diffusion restoration models. *CoRR*, abs/2209.11888, 2022.

Kawar, B., Ganz, R., and Elad, M. Enhancing diffusion-based image synthesis with robust classifier guidance. In *Trans. Mach. Learn. Res. (TMLR)*, 2023.

Kim, M. and Leskovec, J. The network completion problem: Inferring missing nodes and edges in networks. In *Proc. Int. Conf. on Data Mining (SDM)*, 2011.

Kipf, T. N. and Welling, M. Variational graph auto-encoders. *CoRR*, abs/1611.07308, 2016.

Kulikov, V., Yadin, S., Kleiner, M., and Michaeli, T. Sinddm: A single image denoising diffusion model. In *Proc. Int. Conf. on Machine Learning, (ICML)*, 2023.

Leskovec, J. and Faloutsos, C. Sampling from large graphs. In *Proc. Int. Conf. on Knowledge Discovery & Data Mining (KDD)*, 2006.

Liao, R., Li, Y., Song, Y., Wang, S., Nash, C., Hamilton, W. L., Duvenaud, D., Urtasun, R., and Zemel, R. Efficient graph generation with graph recurrent attention networks. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2019.

Limnios, S., Selvaraj, P., Cucuringu, M., Maple, C., Reinert, G., and Elliott, A. Sagess: Sampling graph denoising diffusion model for scalable graph generation. *CoRR*, 2023.

Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., and Gool, L. V. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proc. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Luhman, T. and Luhman, E. Improving diffusion model efficiency through patching. *CoRR*, abs/2207.04316, 2022.

Mahdi Karami, Igor Krawczuk, V. C. Multi-resolution graph diffusion. In *ICLR 2024 Workshop on Machine Learning for Genomics Explorations*, 2024.

Manvel, B. On reconstructing graphs from their sets of subgraphs. *J. Comb. Theory, Ser. B*, 21(2), 1976.

Martinkus, K., Loukas, A., Perraudin, N., and Wattenhofer, R. SPECTRE: spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2022.

McGregor, A. and Sengupta, R. Graph reconstruction from random subgraphs. In *Int. Coll. on Automata, Languages, and Programming (ICALP)*, LIPIcs, 2022.

Newman, M. E. J. *Networks: An Introduction*. Oxford University Press, 2010.

Nie, W., Guo, B., Huang, Y., Xiao, C., Vahdat, A., and Anandkumar, A. Diffusion models for adversarial purification. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2022.

Nikankin, Y., Haim, N., and Irani, M. Sinfusion: Training diffusion models on a single image or video. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2023.

Niu, C., Song, Y., Song, J., Zhao, S., Grover, A., and Ermon, S. Permutation invariant graph generation via score-based generative modeling. In *Proc. Int. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2020.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., and Sutskever, I. Zero-shot text-to-image generation. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2021.

Rendsburg, L., Heidrich, H., and von Luxburg, U. Netgan without GAN: from random walks to low-rank approximations. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2020.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models. In *Proc. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Saharia, C., Chan, W., Chang, H., Lee, C. A., Ho, J., Salimans, T., Fleet, D. J., and Norouzi, M. Palette: Image-to-image diffusion models. In *Proc. Special Interest Group on Computer Graphics and Interactive Techniques Conf. (SIGGRAPH)*, 2022a.

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E. L., Ghasemipour, S. K. S., Lopes, R. G., Ayan, B. K., Salimans, T., Ho, J., Fleet, D. J., and Norouzi, M. Photorealistic text-to-image diffusion models with deep language understanding. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2022b.

Seshadhri, C., Sharma, A., Stolman, A., and Goel, A. The impossibility of low-rank representations for triangle-rich complex networks. In *Proc. Natl. Acad. Sci. USA*, 2020.

Simonovsky, M. and Komodakis, N. Graphvae: Towards generation of small graphs using variational autoencoders. In *Proc. Int. Conf. on Artificial Neural Networks (ICANN)*, 2018.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2015.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2019.

Song, Y., Garg, S., Shi, J., and Ermon, S. Sliced score matching: A scalable approach to density and score estimation. In *Proc. Conf. on Uncertainty in Artificial Intelligence (UAI))*, 2019.

Vignac, C., Krawczuk, I., Siraudin, A., Wang, B., Cevher, V., and Frossard, P. Digress: Discrete denoising diffusion for graph generation. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2023.

Wang, H., Wang, J., Wang, J., Zhao, M., Zhang, W., Zhang, F., Xie, X., and Guo, M. Graphgan: Graph representation learning with generative adversarial nets. In *Proc. Conf. on Adv. of Artificial Intelligence (AAAI)*, 2018.

Wang, H., Yin, H., Zhang, M., and Li, P. Equivariant and stable positional encoding for more powerful graph neural networks. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2022.

Wang, Z., Jiang, Y., Zheng, H., Wang, P., He, P., Wang, Z., Chen, W., and Zhou, M. Patch diffusion: Faster and more data-efficient training of diffusion models. *arXiv preprint arXiv:2304.12526*, 2023.

Xiao, Z., Kreis, K., and Vahdat, A. Tackling the generative learning trilemma with denoising diffusion gans. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2022.

Xie, Y., Shi, C., Zhou, H., Yang, Y., Zhang, W., Yu, Y., and Li, L. MARS: markov molecular sampling for multi-objective drug discovery. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2021.

Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., Shao, Y., Zhang, W., Yang, M., and Cui, B. Diffusion models: A comprehensive survey of methods and applications. *CoRR*, abs/2209.00796, 2022.

Ying, Z., Bourgeois, D., You, J., Zitnik, M., and Leskovec, J. Gnnexplainer: Generating explanations for graph neural networks. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2019.

You, J., Ying, R., Ren, X., Hamilton, W. L., and Leskovec, J. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2018.

You, J., Ying, R., and Leskovec, J. Position-aware graph neural networks. In *Proc. Int. Conf. on Machine Learning (ICML)*, 2019.

Zahirnia, K., Schulte, O., Nadaf, P., and Li, K. Micro and macro level graph modeling for graph variational auto-encoders. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2022.

Zhang, M. and Chen, Y. Link prediction based on graph neural networks. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2018.

Zhang, M., Li, P., Xia, Y., Wang, K., and Jin, L. Revisiting graph neural networks for link prediction. In *Proc. Adv. in Neural Information Processing Systems (NeurIPS)*, 2020.

Zhang, Q. and Chen, Y. Fast sampling of diffusion models with exponential integrator. In *Proc. Int. Conf. on Learning Representations (ICLR)*, 2023.

Zhang, Q., Song, J., Huang, X., Chen, Y., and yu Liu, M. Diffcollage: Parallel generation of large content with diffusion models. In *Proc. Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Zhu, Y., Du, Y., Wang, Y., Xu, Y., Zhang, J., Liu, Q., and Wu, S. A survey on deep graph generation: Methods and applications. In *Proc. Conf. on Learning on Graphs (LOG)*, 2022.

A. Appendix

A.1. Denoising Results

We include our detailed results for denoising here. EDGE uses the degree sequence to help guide the reverse process of large scale graphs. Here, we include a "cheatcode" that uses the "target" degree and observed subgraph (instead of the observed degree sequence). In some sense, this provides the complete information needed to recover the unobserved target graph.

Table 5. **Graph Denoising, BA-Shapes.**

Method	**SGDM?**	**PE?**	**Conditioning?**	**Consensus** (↑)	**Diversity** (↑)	**Sparsity** (≈ 1)	**Edge Overlap** (≈ 1)
EDGE	✓	✓	Obs. Degree + Subst.	0.256	0.903	0.874	0.8212
EDGE	✓	✓	Target Degree + Subst.	0.580	0.976	0.797	0.753
EDGE	✓	✓	Obs. Degree + Subst.	0.341	0.928	0.762	0.721
EDGE	✓	✓	Target Degree + Subst.	0.544	0.979	0.720	0.6838
EDGE	✗	✗	Obs. Degree + Subst.	0.954	0.834	0.055	0.04273
EDGE	✗	✗	Target Degree + Subst.	0.950	0.852	0.063	0.04562
EDGE	✗	✓	Obs. Degree + Subst.	0.996	0.733	0.019	1.519
EDGE	✗	✓	Target Degree + Subst.	0.996	0.736	0.020	1.560
GDSS	✓	✗	Subst.	0.098	0.979	0.373	0.58165
GDSS	✓	✓	Subst.	0.023	0.825	0.578	0.89488
DiGRESS	✓	✓	Subst.	0.184	0.589	0.914	–
DiGRESS	✓	✓	Subst.	0.007	0.239	1.016	–
DiGRESS	✓	✓	No Subst.	0.016	0.421	1.026	–

Table 6. **Graph Denoising, CORA.**

Method	**SGDM?**	**PE?**	**Conditioning?**	**Consensus** (↑)	**Diversity** (↑)	**Sparsity** (≈ 1)	**Edge Overlap** (≈ 1)
EDGE	✓	✗	Obs. Degree + Subst.	0.587	1.000	0.591	0.58509
EDGE	✓	✗	Target Degree + Subst.	0.731	0.999	0.588	0.58188
EDGE	✓	✓	Obs. Degree + Subst.	0.754	1.000	0.442	0.43763
EDGE	✓	✓	Target Degree + Subst.	0.836	1.000	0.441	0.43677
EDGE	✗	✗	Obs. Degree + Subst.	0.916	0.915	0.115	0.09570
EDGE	✗	✗	Target Degree + Subst.	0.873	0.931	0.118	0.09387
EDGE	✗	✓	Obs. Degree + Subst.	0.965	0.798	0.057	0.04864
EGDE	✗	✓	Target Degree + Subst.	0.978	0.999	0.067	0.05109
GDSS	✓	✓	Subst.	0.029	0.723	0.970	0.92559
GDSS	✓	✗	Subst.	0.619	1.000	0.322	0.30592
DiGRESS	✓	✓	None	0.402	0.998	2.546	–
DiGRESS	✓	✓	Subst.	0.438	0.874	0.865	0.82835

Table 7. **Graph Denoising, PolBlogs.**

Method	**SGDM?**	**PE?**	**Conditioning?**	**Consensus** (↑)	**Diversity** (↑)	**Sparsity** (≈ 1)	**Edge Overlap** (≈ 1)
EDGE	✓	✗	Obs. Degree + Subst.	0.713	1.000	0.340	0.31515
EDGE	✓	✗	Target Degree + Subst.	0.605	1.000	0.437	0.4044
EDGE	✓	✓	Obs. Degree + Subst.	0.375	1.000	0.640	0.59266
EDGE	✓	✓	Target Degree + Subst.	0.469	1.000	0.686	0.6356
EDGE	✗	✓	Obs. Degree + Subst.	0.993	0.598	0.023	0.02181
EDGE	✗	✓	Target Degree + Subst.	0.991	0.539	0.018	0.0170
EDGE	✗	✗	Obs. Degree + Subst.	0.992	0.702	0.0411	0.0412
EDGE	✗	✗	Target Degree + Subst.	0.992	0.717	0.042	0.03972
GDSS	✓	✗	Subst.	0.683	1.000	0.353	0.31975
GDSS	✓	✓	Subst.	0.010	0.605	1.078	0.97685
DiGRESS	✓	✓	Subst.	0.8239	0.99603	0.60938	–
DiGRESS	✓	✓	No Subst.	0.886	1.000	2.125	0.42513

A.2. Expansion Results

We include our detailed results for expansion here. EDGE uses the degree sequence to help guide the reverse process of large scale graphs. Here, we include a "cheatcode" that uses the "target" degree and observed subgraph (instead of the observed degree sequence). In some sense, this provides the complete information needed to recover the unobserved target graph.

Table 8. **Graph Expansion, BA-Shapes.**

Method	**SGDM?**	**PE?**	**Conditioning?**	**Consensus** (↑)	**Diversity** (↑)	**Sparsity** (≈ 1)	**Edge Overlap** (≈ 1)
EDGE	✗	✓	Obs. Degree + Subst	0.000	0.100	0.930	1.000
EDGE	✗	✓	Target Degree + Subst.	0.044	0.618	0.958	1.000
EDGE	✗	✗	Obs. degree + Subst.	0.000	0.100	0.930	1.000
EDGE	✗	✗	Target Degree + Subst.	0.030	0.598	0.960	1.000
EDGE	✓	✗	Obs. Degree + Subst.	0.055	0.100	<u>0.930</u>	1.000
EDGE	✓	✗	Target Degree + Subst.	0.589	0.511	1.002	1.000
EDGE	✓	✓	Obs. Degree + Subst.	0.055	0.100	<u>0.930</u>	1.000
EDGE	✓	✓	Target Degree + Subst.	0.594	0.511	1.002	1.000
GDSS	✓	✗	Subst.	0.296	0.983	1.707	1.000
GDSS	✓	✓	Subst.	0.869	0.565	8.798	1.000
DiGRESS	✓	✓	Subst.	<u>0.309</u>	0.616	**1.039**	1.000
DiGRESS	✓	✓	None	**0.550**	**1.000**	5.738	1.000

Table 9. **Graph Expansion, CORA.**

Method	**SGDM?**	**PE?**	**Conditioning?**	**Consensus** (↑)	**Diversity** (↑)	**Sparsity** (≈ 1)	**Edge Overlap** (≈ 1)
EDGE	✗	✗	Obs. Degree + Subst.	0.000	0.100	0.922	1.00
EDGE	✗	✗	Target Degree + Subst.	0.020	0.520	0.944	1.00
EDGE	✗	✓	Obs. Degree + Subst.	0.000	0.100	0.922	1.000
EDGE	✗	✓	Target Degree + Subst.	0.020	0.493	0.940	1.000
EDGE	✓	✗	Obs. Degree + Subst.	0.000	0.100	0.925	1.00
EDGE	✓	✗	Target Degree + Subst.	0.550	0.510	1.001	1.00
EDGE	✓	✓	Obs. Degree + Subst.	0.000	0.100	0.918	1.000
EDGE	✓	✓	Target Degree + Subst.	0.542	0.535	1.000	1.000
GDSS	✓	✗	Subst.	0.246	1.000	5.098	1.000
GDSS	✓	✓	Subst.	0.223	1.000	10.730	1.000
DiGRESS	✓	✓	Subst.	0.117	0.933	1.831	1.000
DiGRESS	✓	✓	None	0.142	0.968	1.755	1.000

Table 10. **Graph Expansion, PolBlogs.**

Method	**SGDM?**	**PE?**	**Conditioning?**	**Consensus** (↑)	**Diversity** (↑)	**Sparsity** (≈ 1)	**Edge Overlap** (≈ 1)
EDGE	✗	✗	Obs. Degree + Subst.	0	0.100	0.941	1.000
EDGE	✗	✗	Target Degree + Subst.	0.046	0.909	0.998	1.000
EDGE	✗	✓	Obs. Degree + Subst.	0	0.100	0.938	1.000
EDGE	✗	✓	Target Degree + Subst.	0.046	0.888	0.994	1.000
EDGE	✓	✗	Obs. Degree + Subst.	0.000	0.100	0.939	1.000
EDGE	✓	✗	Target Degree + Subst.	0.681	0.363	1.002	1.000
EDGE	✓	✓	Obs. Degree + Subst.	0.000	0.100	0.939	1.000
EDGE	✓	✓	Target Degree + Subst.	0.652	0.392	1.002	1.000
GDSS	✓	✗	Subst.	0.217	1.000	5.444	1.000
GDSS	✓	✓	Subst.	0.305	1.000	14.174	1.000
DiGRESS	✓	✓	Subst.	0.034	0.991	1.153	1.000
DiGRESS	✓	✓	None	0.031	0.994	1.134	1.000

A.3. Style Transfer Results

Table 11. **Graph Style Transfer**

Task	**PE?**	**Regressor Type**	**MAE**(↓)
Sum-Degree	✗	GCN	6.46
	✗	GIN	6.81
	✗	GTrans	6.48
	✓	GCN	3.78
	✓	GIN	3.62
	✓	GTrans	3.45
Max-Degree	✗	GCN	24.73
	✗	GIN	45.09
	✗	GTrans	24.63
	✓	GCN	9.3
	✓	GIN	11.04
	✓	GTrans	11.44
Num. Triangles	✗	GCN	9.18
	✗	GIN	9.57
	✗	GTrans	10.29
	✓	GCN	19.6
	✓	GIN	19.6
	✓	GTrans	19.6

A.4. Experimental Figures

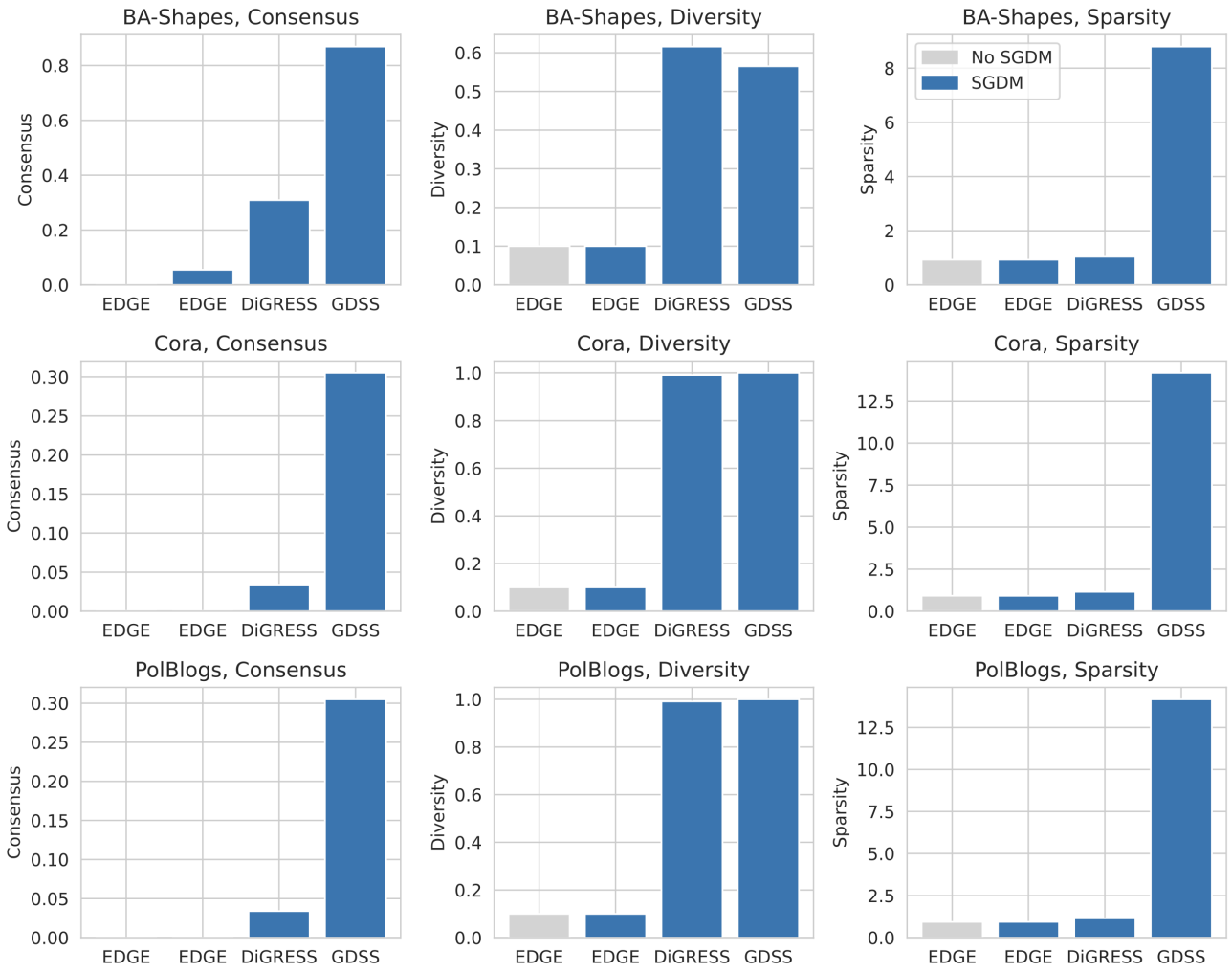

Figure 3. **Effect of Diffusion Backbone on Denoising**

A.5. Generating Large Graphs with Subgraph-based Diffusion

Here, we demonstrate that our proposed subgraph-based diffusion framework performs comparably with other generative graph models. We use a simple "stitching" based algorithm: (1) we generate an ego-graph around a particular node, and (2) stitches the subgraphs together. Positional encodings are used in lieu of node-ids as the positional encodings of additional nodes can be approximated using perturbation theory.

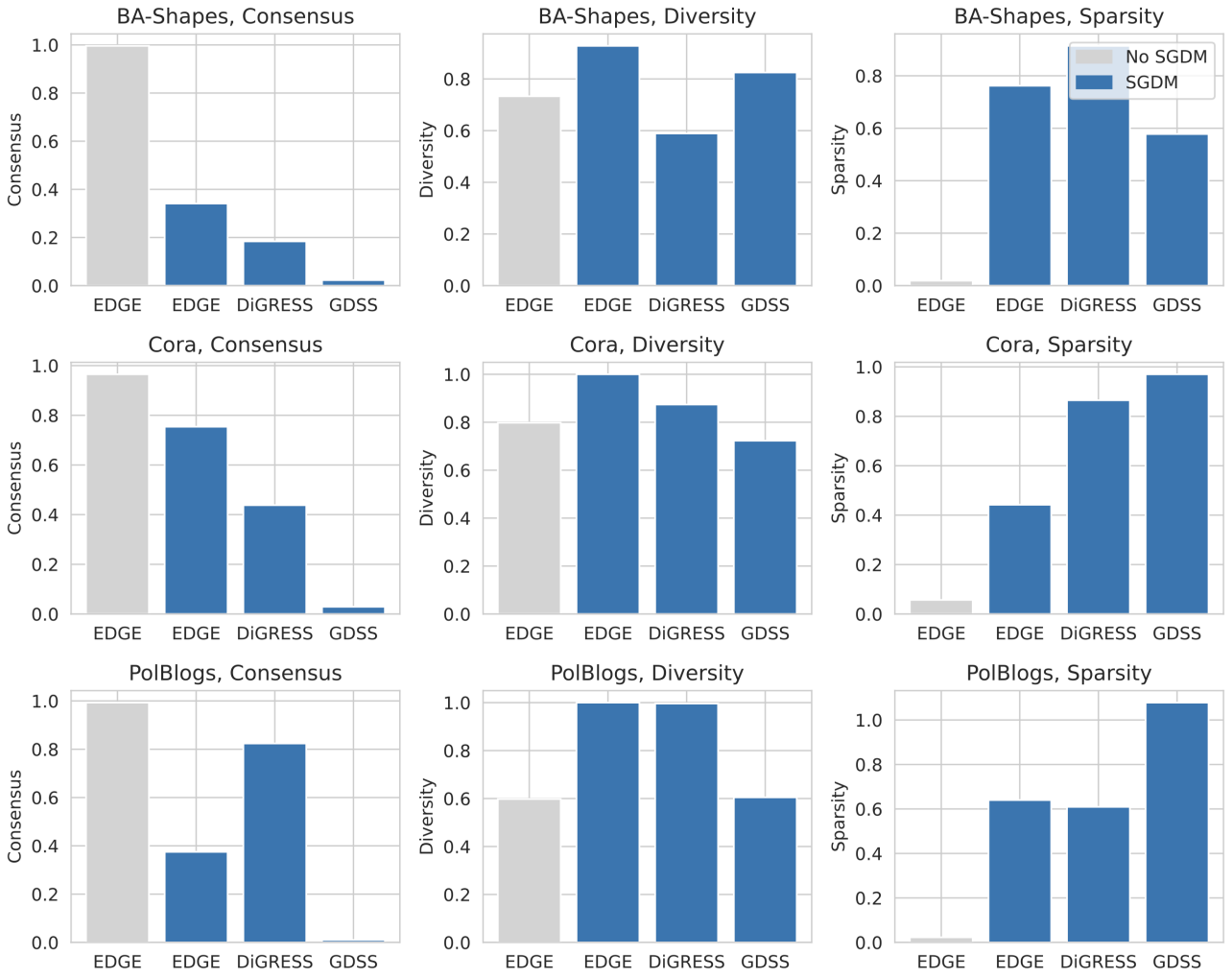

Figure 4. **Effect of Diffusion Backbone on Expansion.** Not that while it appears that No SGDM-EDGE has better consensus, note that it also has significantly lesser sparsity.

A.6. Expanded Related Work

Here, we expand on the related work introduced in Sec. 2. In brief, diffusion-based generative models consist of two parts: (i) a *forward* process, in which data, $\boldsymbol{x}_0 \sim q(\boldsymbol{x})$, is gradually transformed to a reference distribution through the addition of some noise over time ϵ_t, and (ii) *reverse* process, which takes samples from the reference distribution and attempts to denoise the samples over time to synthesize samples from the original data distribution.

Early graph diffusion models closely followed the formulation of diffusion models for image synthesis, and ignored the discreteness of graph data when designing these processes. For example, Niu et al. (2020) use a multivariate standard Gaussian as the reference distribution and add Gaussian noise to the adjacency matrix when the performing the forward step. After the final reverse step is completed, the continuous graph is discretized. While their discrete-time score-based formulation leads to permutation invariant generation (e.g., different node permutations of a graph have the same probability), intermediate representations are continuous, reducing their interpretability and requiring 1000+ steps to generate good samples. Jo et al. (2022) propose a continuous-time system of stochastic differential equations to jointly capture the dependency of nodes and graph topology. However, since the forward process still adds Gaussian noise to the adjacency matrix, it also suffers from cumbersome sampling.

Algorithm 3 Large Graph Generation with **SGDM**

Input: p_θ:trained reverse process, `Hist-Global`: distribution of observed global context, `Hist-GraphSize`: distribution of observed subgraph sizes
Output: $\mathcal{G}_G = (\mathbf{A}_G, \mathbf{X}_G, \mathbf{C}_G, \mathbf{Q}_G)$: generated graph
$\boldsymbol{\mathcal{S}} = \{\}$
while $\exists C_i \in \mathbf{C}$ *not SAMPLED* **do**
  $n \leftarrow$ `SAMPLE(Hist-GraphSize)`
  $\mathbf{C}_{\mathcal{S}_T} \leftarrow$ `SAMPLE(Hist-Global,` n`)`
  $\mathcal{S}^{t=T} \leftarrow q_n(\cdot)$
  for $t = T$ **to** 0 **do**
    $\mathbf{Q}^t \leftarrow$ `Local-Context(`$\mathcal{S}^t$`)`
    $\mathcal{S}^t \leftarrow \mathbf{Q}^t$
    $\mathcal{S}^{t-1} \leftarrow p_\theta(\mathcal{S}^t, \mathbf{C})$
    $(\mathbf{A}^{t-1}, \mathbf{X}^{t-1}, \mathbf{C}, \mathbf{Q}^t) \leftarrow \mathcal{S}^{t-1}$
  end
  $\boldsymbol{\mathcal{S}} \leftarrow \mathcal{S}^{t=0}$
end
▷ `Use Global-Context to Merge Subgraphs.`
$\mathcal{S}_G =$ `COALESCE(`$\boldsymbol{\mathcal{S}}$`)`
return $\mathcal{S}_G$

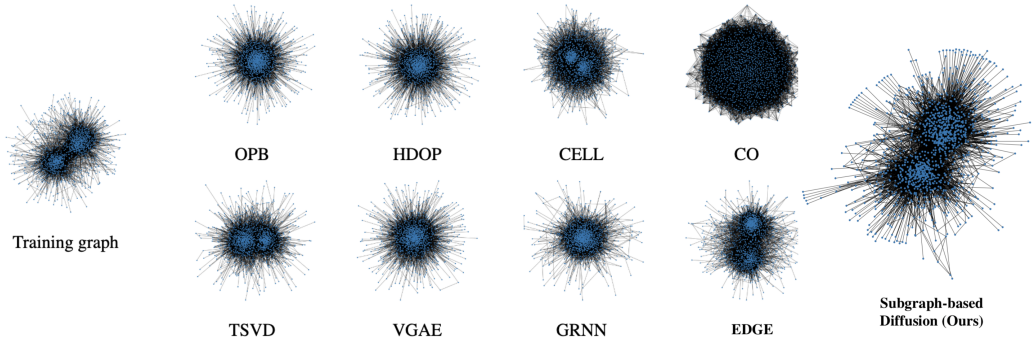

Figure 5. **Example of Stitching.** Compared against several other large-scale generative models, including auto-encoder, recurrent and diffusion based models, we see that our proposed subgraph-based diffusion framework is able to generate graphs qualitatively similar graphs to the training sample. Visualization of other methods is reproduced from (Chen et al., 2023).

A Gaussian reference distribution and noising process was well-suited for image synthesis, as it not only enabled tractable formulations of the forward/reverse processes, but also supported natural image inductive biases. Namely, that adding Gaussian noise in the forward process will remove high frequency components (style) before low frequency components (content), allowing models to learn to "paint" pictures. Because graphs are discrete, structured data, Gaussian noise/priors are not as applicable since they destroy the sparsity and alter topological properties, leading to subpar performance. Therefore, more recent graph diffusion models have focused on the discrete diffusion processes (Austin et al., 2021; Hoogeboom et al., 2021).

Instead of using multivariate Gaussian's as the reference distribution and Gaussian noise during the forward process, Vignac et al. (2023) uses a uniform distribution over different types of edge and nodes types as the reference distribution and performs edge/node edits to simulate changing the particular edge/node category. Since the reference distribution is over the categorical edge/node, they cast the training objective as correctly classifying the edge/node type as each time step, which is much simpler than learning each node/edge. Moreover, they find that defining the transition probabilities in terms of the marginal distribution over the edge/node anchors observed in the training distribution (instead of uniform) can substantially decrease the number of denoising steps and improve the overall generation quality. Lastly, to circumvent limitations in GNN expressiveness, DIGRESS leverages the discrete intermediate steps to compute structural features at each time step. However, we note that DiGRESS is expensive to train on large graphs and expects only categorical edges/nodes, which can be limiting for real-datasets.

In order to improve the scalability of graph diffusion models, while retaining high-fidelity, Chen et al. (2023) (EDGE)

Table 12. **Large Graph Generation using Subgraph Diffusion.** On the PolBlogs dataset, we compare the performance of our proposed subgraph diffusion framework against several large graph generation methods. The performance of other generative models is reproduced from (Chen et al., 2023). Better performance is obtained models which produce graphs whose statistics are closer to the training graph, without having memorized the training graph (Edge Overlap $<<$ 100).

	Edge Overlap	**Power Law Exp.**	**Num. Triangle**	**Clustering Coeff.**	**Characteristic Path Length**	**Assortativity Coeff.**
Training Graph	100	1.414	1	0.226	2.738	-0.221
OPB (Chanpuriya et al., 2021)	24.5	1.395	0.667	0.150	2.524	-0.143
HDOP (Chanpuriya et al., 2021)	16.4	1.393	0.687	0.153	2.522	-0.131
CELL (Rendsburg et al., 2020)	26.8	1.385	0.810	0.211	2.534	-0.230
CO (Chanpuriya et al., 2021)	20.1	1.975	0.045	0.028	2.502	0.068
TSVD (Seshadhri et al., 2020)	32.0	1.373	0.872	0.205	2.532	-0.216
VGAE (Kipf & Welling, 2016)	3.6	1.723	0.05	0.001	2.531	-0.086
GRNN (You et al., 2018)	9.6	1.333	0.354	0.095	2.566	0.096
EDGE (Chen et al., 2023)	16.5	1.398	0.977	0.217	2.647	-0.214
SGDM(Ours)	9.44	1.384	1.301	0.308	2.612	-0.264

proposes an alternative discrete denoise process, which uses an empty graph as the reference distribution and defines the forward process as sequentially removing edges to an empty state. In order to effectively scale to large graphs, EDGE limits the number of "active" nodes by only conditioning the propagation step on whether a nodes degree has changed from t to $t-1$. Furthermore, since the degree distribution is closely related to many graph properties, EGDE conditions on the degree distribution (since they use a discrete process), a tangential benefit of their discrete process. Jo et al. (2023) (DRuM) learns distribution over possible trajectories using Schrodinger's Bridges, instead of the a single point, in order to ensure that intermediate steps are well-aligned with the true data-distribution. Lastly, we note that our work is contemporary to Limnios et al. (2023), which also uses sampling and reconstruction to improve the scalability of diffusion models (in particular DiGress). However, there are several differences: namely, SGDM is a framework that is flexible to the choice of underlying diffusion model, uses both global/local contexts to improve the performance, and does not require discrete node feature ids to perform the reconstruction step. Furthermore, in addition to SGDM, we also formalize several network editing tasks and demonstrate that SGDM can be used to effectively perform these tasks.

Table 13. **SAMPLED Dataset Statistics.** After performing 2-hop ego net sampling, we obtain the following dataset statistics. Note, that while the average and median number of edges is large for PolBlogs, we perform SUBSAMPLING to control this size. Furthermore, note that since average graph size is now more mangeable, we can also perform batching to reduce training time.

Dataset	Num. Graphs	Avg. Nodes	Med. Nodes	Avg. Edges	Med. Edges
BA-Shapes	700	159	91	1,300	502
PolBlogs	1,222	486	519	15,768	17,129
CORA	2,485	139	92	524	296

A.7. Experimental Details

We trained all models using Tesla T4s (16GB GPU Memory, 124GB RAM). To ensure fair comparison across methods and prevent overfitting to a corrupted graph, all models are trained for at most 24 hours or 5000 epochs, which ever came first. All metrics are reported over 10 generated samples. When computing edge overlap, we first sort the graph by the degree sequence and then compare.

We use the official released code by DiGRESS (https://github.com/cvignac/DiGress), EDGE (https://github.com/tufts-ml/graph-generation-EDGE) and GDSS (https://github.com/harryjo97/GDSS). Our evaluation set-up, metrics and editing tasks were implemented using PyTorch Geometric (Fey & Lenssen, 2019) and PyTorch. Our code can be accessed at https://github.com/pujacomputes/sgdm.

For the SAMPLING function, we use 2-hop ego networks and, for SUBSAMPLING, perform uniform randomly subsampling to obtain subgraphs of maximum graph-size 50 for training. For GLOBAL-CONTEXT, we use the top-two non-zero eigenvectors of $\mathcal{G}_T$'s Laplacian. Hyper-parameters and architectures suggested by each method's authors are used. However, to incorporate global context, we use the following strategy, inspired by how vision diffusion models incorporate positional coordinates (Luhman & Luhman, 2022; Wang et al., 2023): (i) we project the global context vector to the same size of the node's hidden representation vector (ii) at each message passing layer, we add the global context projection to each node representation.

As discussed in Sec. 2 and Sec. A.6, EDGE uses the degree sequence to help guide large models in the reverse process. We include additional refinement results that show the performance of EDGE when using the observed subgraph and the *target* degree sequence. This can be seen as giving the reverse process a "cheat-code", since all the information to recover the unobserved, clean subgraph has been provided. We note that using EDGE directly with the observed subgraph's degree distribution (without global-context) requires that we perform graph-alignment with respect to the observed subgraph. Using global context with the observed degree helps avoid this alignment issue for feature-less graphs as ids can be matched using the global context. While we directly filled in (removed) edges when working with DiGRESS, GDSS uses a real-valued adjacency matrix. Therefore, when performing expansion, we would fill in the maximum value at a given time t for the edges that we wished to condition upon. Lastly, we note that while our expansion task definition appears similar to link prediction, this task and the discuss in-painting approach also provides support for filling in missing features. Since EDGE does not support features at this time, we only present results for subgraph denoising.

A.8. Alternative Instantiations

We used k-hop ego networks for the SAMPLING functions as we desired a node-centric sampling scheme and induce subgraphs from a randomly selected subset of larger graphs to perform SUBSAMPLING. We were motivated from the perspective that particular nodes may be unreliable and lead to corruptions in the observed graph. For example, we can imagine malicious users in a social network. Given that the majority of nodes (and their surrounding neighborhoods) are not corrupted and nodes (users) are expected to have some similar behaviors, learning a distribution over the induced ego-centric subgraphs will allow information transfer between the majority, uncorrupted nodes and the corrupted ones.

While we use random subsampling and ego-nets in our experiments, alternatives are also possible (Leskovec & Faloutsos, 2006). For example, instead of effectively doing Breadth First Search, we can also do random walk sampling, forest fire sampling, or potentially node2vec style sampling starting from each node in the observed graph so that we can balance the amount of global network structure or local network structure captured in each subgraph. While the above provide different strategies for generating node-centric graphs, we can also define a SAMPLING function that is not node-centric but still satisfies the criteria introduced in Sec. 4. For example, we can use METIS to obtain a non-overlapping graph partition and extend each partition by 1-hop to make sure to share information across partitions. The choice of SAMPLING function helps control how many times a single node is seen over the collection of all subgraphs.

We note that the choice of SUBSAMPLING function is also flexible in **SGDM**. While we used uniform node sampling for its ease of use, it is also possible to perform SUBSAMPLING with other strategies. For example, we can use a distribution biased by node degree instead of a uniform distribution. Lastly, we note that there are several choices for GLOBAL-CONTEXT as well. For example, instead of Laplacian Positional vectors, we can use position aware embeddings (You et al., 2019), which computes a distance between a node and a set of anchor nodes.

A.9. Additional Experimental Results

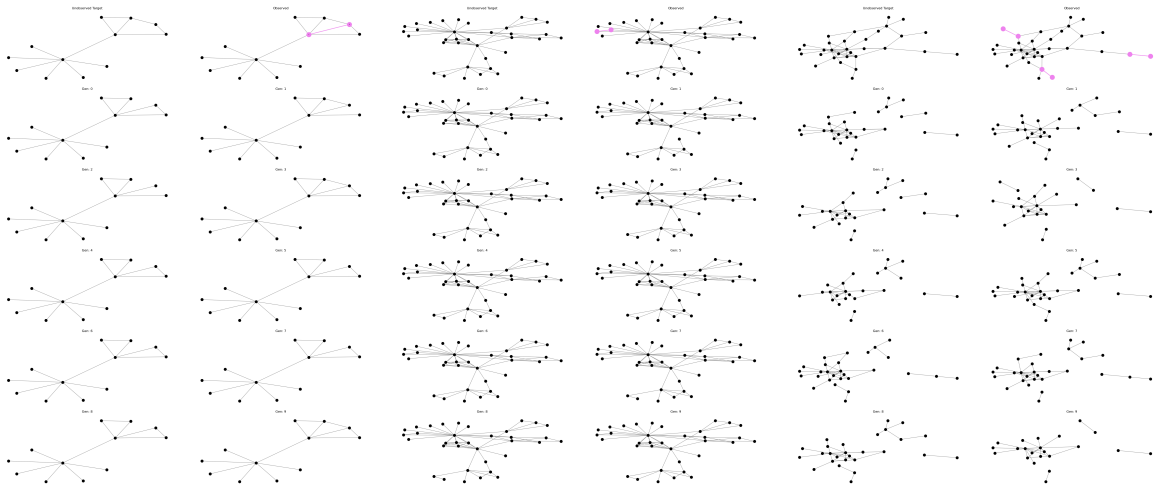

Figure 6. **Digress, (BAShapes, Cora, PolBlogs) Denoising.**

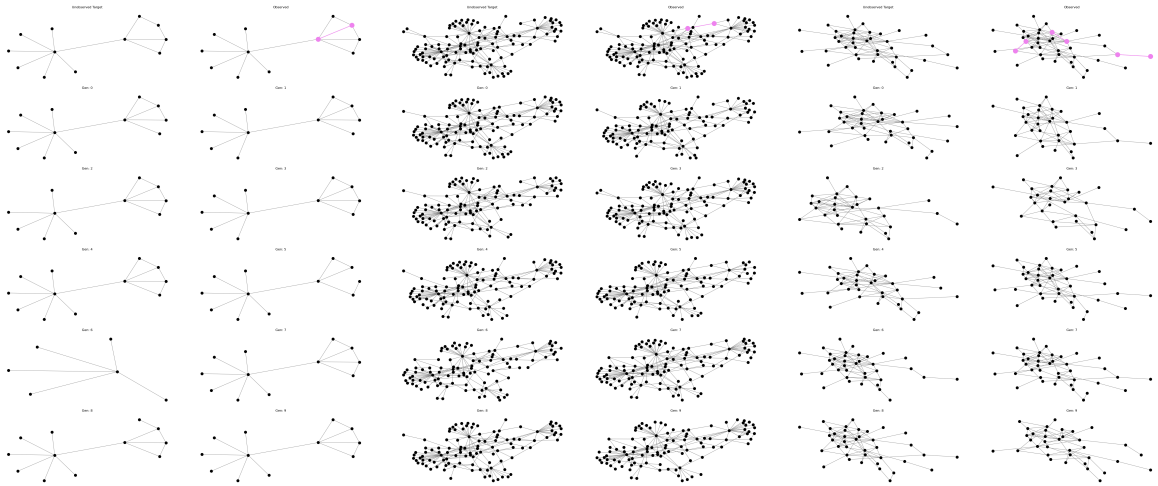

Figure 7. **EDGE, (BAShapes, Cora, PolBlogs) Denoising.**