arXiv:2408.16457v6 [cs.LG] 17 Aug 2026

HYGENE: A Diffusion-based Hypergraph Generation Method

Dorian Gailhard, Enzo Tartaglione, Lirida Naviner, Jhony H. Giraldo

LTCI, Télécom Paris, Institut Polytechnique de Paris, France
{name.surname}@telecom-paris.fr

Abstract

Hypergraphs are powerful mathematical structures that can model complex, high-order relationships in various domains, including social networks, bioinformatics, and recommender systems. However, generating realistic and diverse hypergraphs remains challenging due to their inherent complexity and lack of effective generative models. In this paper, we introduce a diffusion-based **Hy**pergraph **Gene**ration (HYGENE) method that addresses these challenges through a progressive local expansion approach. HYGENE works on the bipartite representation of hypergraphs, starting with a single pair of connected nodes and iteratively expanding it to form the target hypergraph. At each step, nodes and hyperedges are added in a localized manner using a denoising diffusion process, which allows for the construction of the global structure before refining local details. Our experiments demonstrated the effectiveness of HYGENE, proving its ability to closely mimic a variety of properties in hypergraphs. To the best of our knowledge, this is the first attempt to employ diffusion models for hypergraph generation. Our code is open source[1].

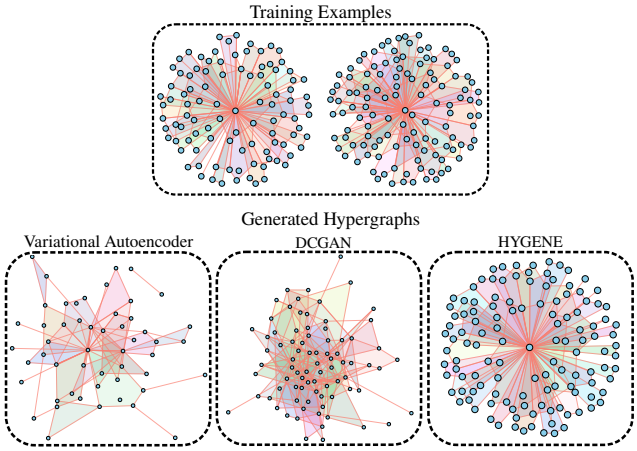


Figure 1: Examples of generated ego hypergraphs by a variational autoencoder, a Deep Convolutional Generative Adversarial Network (DCGAN), and our model (HYGENE).

1 Introduction

Hypergraphs are higher-order extensions of graphs. They comprise a set of nodes, also called vertices, and a set of hyperedges. Unlike regular graphs, where edges connect only two nodes, hyperedges can connect any number of nodes. These structures have demonstrated their ability to capture more complex relationships than graphs and have been applied in various domains (Gong, Higham, and Zygalakis 2023; Estrada and Rodríguez-Velázquez 2006). For instance, hypergraphs have been applied in drug discovery (Kajino 2019), modeling contagion spread (Higham and De Kergorlay 2021), and electronics (Starostin and Balashov 2008; Luo et al. 2024; Grzesiak-Kopeć, Oramus, and Ogorzałek 2017). Besides, they have also proven useful in recommender systems (Lin et al. 2023), molecular biology (Murgas, Saucan, and Sandhu 2022; Rahman et al. 2012), and urban planning (Dupagne and Teller 1998). The versatility of hypergraphs in representing multi-way relationships makes them a powerful tool across these diverse fields; consequently, hypergraph generation (the ability to sample from specific hypergraph distributions) holds significant promise.

Despite its wide applicability, research in hypergraph generation has primarily focused on algorithmic approaches, aiming to develop methodologies that produce hypergraphs with specific, predefined structural properties (Do et al. 2020; Arafat et al. 2020). In contrast, the exploration of hypergraph generation using deep learning models remains largely understudied. This gap represents a significant opportunity to advance this field since deep learning approaches may capture complex patterns and generate more realistic and diverse hypergraphs. Such methods could enhance the modeling of intricate relationships beyond the scope of traditional graph structures.

In contrast to hypergraph generation, deep learning-based graph generation has been extensively studied (Zhu et al. 2022). Learning-based graph generation can be broadly categorized into two approaches: *one-shot* approaches that generate the entire graph simultaneously (Simonovsky and Komodakis 2018; You et al. 2018; Vignac et al. 2023), and *iterative* models that generate the graphs incrementally, predicting edges for each new node (Chen et al. 2023; Bergmeister

Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1]https://github.com/DorianGailhard/HYGENE

et al. 2024). While graph generation techniques have shown promise, their adaptation to hypergraphs remains challenging. The variable size of hyperedges and their higher-order relationships increase the difficulty of the task, making direct application of graph methods non-trivial. Figure 1 shows that naïvely generating the incidence matrix using classical image generation architectures is not sufficient either, as it lacks the correct understanding of the underlying data structure.

By leveraging the spectral equivalence between a hypergraph and two carefully chosen representations (the clique and star expansions), we generalize the work by Bergmeister et al. (2024) for hypergraph generation. Their method is based on an iterative local expansion scheme, where graph generation is performed hierarchically, first building the global structure before refining the details. This process is seen as the inverse of a coarsening operation, which involves the reduction of a graph while preserving its relevant properties. For hypergraphs, the variable size of hyperedges increases the complexity of the problem. As every non-empty set of nodes is a possible hyperedge, they are exponentially more numerous than classical edges. In order to mitigate this, we introduce an iterative expansion and refinement process for Hypergraph Generation (HYGENE), rather than predicting all possible hyperedges at once. We train a denoising diffusion model (Karras et al. 2022) using this framework, and validate our method on four synthetic and three real-world datasets, demonstrating its effectiveness in replicating important structural properties. At a glance, our main contributions are the following:

- To the best of our knowledge, we introduce the first diffusion-based method for generating hypergraphs sampled from specific distributions (Sec. 3.2).
- We generalize important concepts in the graph domain to hypergraph generation, like hypergraph coarsening and diffusion (Sec. 3.3, Sec. 3.4, Sec. 3.5).
- We provide rigorous theoretical justifications for our technical choices.
- We validate HYGENE on four synthetic and three real-world datasets, showcasing its ability to capture and reproduce subtle structural properties of hypergraphs (Sec. 4).

2 Related Work

Graph Generation Using Deep Learning. The field of graph generation using deep learning models was pioneered by GraphVAE (Simonovsky and Komodakis 2018). This approach employs an autoencoder to embed graphs into a latent space, from which new graphs could be generated by sampling and decoding. Subsequently, You et al. (2018) significantly enhanced generation quality by using a recurrent neural network to produce the adjacency matrix column-wise. Recent advancements include the work by Kong et al. (2023), which formalized the generation process as an inverse discrete absorption. In this method, nodes from a training graph are sequentially absorbed, and a mixture of multinomials is trained to predict the necessary edge additions when reintroducing a node. Diffusion models, which have shown remarkable success in image generation, were adapted for graph generation in (Niu et al. 2020). This approach was further refined by Vignac et al. (2023) and expanded in (Chen et al. 2023) by incorporating target degree distributions.

A departure from typical methods in graph generation was proposed by Bergmeister et al. (2024). Instead of sequentially adding nodes and predicting their connections, this approach reverses a *coarsening* process. During training, graphs are reduced by merging nodes into clusters. The model then learns to identify these clusters, decompose them, and reconstruct the original connections between their nodes. Graph generation begins with a coarse single-node graph and progressively adds details by expanding it using the trained model, mirroring the diffusion approaches seen in modern image generation techniques. Similar hierarchical concepts have been explored in molecule generation, with Zhu et al. (2023) applying normalizing flows. Related ideas can also be found in the work of Guo, Zou, and Lerman (2023).

In contrast to previous work, we focus on the problem of hypergraph generation, which extends the concept of graph generation to higher-order structures. Furthermore, we employ a hierarchical view of the problem instead of the sequential edge-by-edge generation commonly employed. We also depart from the classical view of edge prediction, where a model outputs the probability of existence for all possible edges, and, instead, predict both the number of hyperedges and their composition.

3 Method

3.1 Preliminaries

Notation. In this work, calligraphic letters such as $\mathcal{V}$ denote sets, and $|\mathcal{V}|$ represents the cardinality of the set. Uppercase boldface letters such as $\mathbf{A}$ represent matrices, while lowercase boldface letters such as $\mathbf{x}$ denote vectors. The superscript $(\cdot)^\mathsf{T}$ corresponds to transposition. $\mathrm{diag}(\mathbf{x})$ denotes a diagonal matrix with entries given by the vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]^\top \in \mathbb{R}^N$. Finally, $\mathrm{Sp}(\mathbf{A})$ denotes the set of eigenvalues of a matrix $\mathbf{A}$.

Basic Definitions. A graph G is defined as a pair $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of vertices and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges. Each edge $e \in \mathcal{E}$ is a pair of vertices (u, v), representing a connection between nodes u and v. A bipartite graph B is a special case of a graph, defined as $(\mathcal{V}_L, \mathcal{V}_R, \mathcal{E})$, where $\mathcal{V}_L$ and $\mathcal{V}_R$ are disjoint sets of vertices, and $\mathcal{E} \subseteq \mathcal{V}_L \times \mathcal{V}_R$. Every edge in a bipartite graph connects a node in $\mathcal{V}_L$ to a node in $\mathcal{V}_R$. Note that a bipartite graph can be identified as a graph where $\mathcal{V} = \mathcal{V}_L \cup \mathcal{V}_R$. We define the Laplacian of a graph $\mathbf{L}_G \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ as $\mathbf{L}_G = \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ is the diagonal degree matrix and $\mathbf{W}$ is the weighted adjacency matrix with $\mathbf{W}_{[i,j]} \neq 0$ if $(i, j) \in \mathcal{E}$. The normalized Laplacian is defined as $\boldsymbol{\mathcal{L}}_G = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$, where $\mathbf{I}$ is the identity.

A hypergraph H is defined as a pair $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of vertices and $\mathcal{E}$ is a set of hyperedges, with each $e \in \mathcal{E}$ being a subset of $\mathcal{V}$. Unlike in graphs, hyperedges can connect any number of vertices. We define two graph representations of hypergraphs: the clique and star expansions. The clique expansion of a hypergraph H is a graph $C = (\mathcal{V}_c, \mathcal{E}_c)$, where $\mathcal{E}_c = \{(u, v) \mid \exists\, e \in \mathcal{E} : u, v \in e\}$. The star expansion of a hypergraph H is a bipartite graph $B = (\mathcal{V}_L, \mathcal{V}_R, \mathcal{E}_b)$, where $\mathcal{V}_L = \mathcal{V}$, $\mathcal{V}_R = \mathcal{E}$, and $\mathcal{E}_b =$

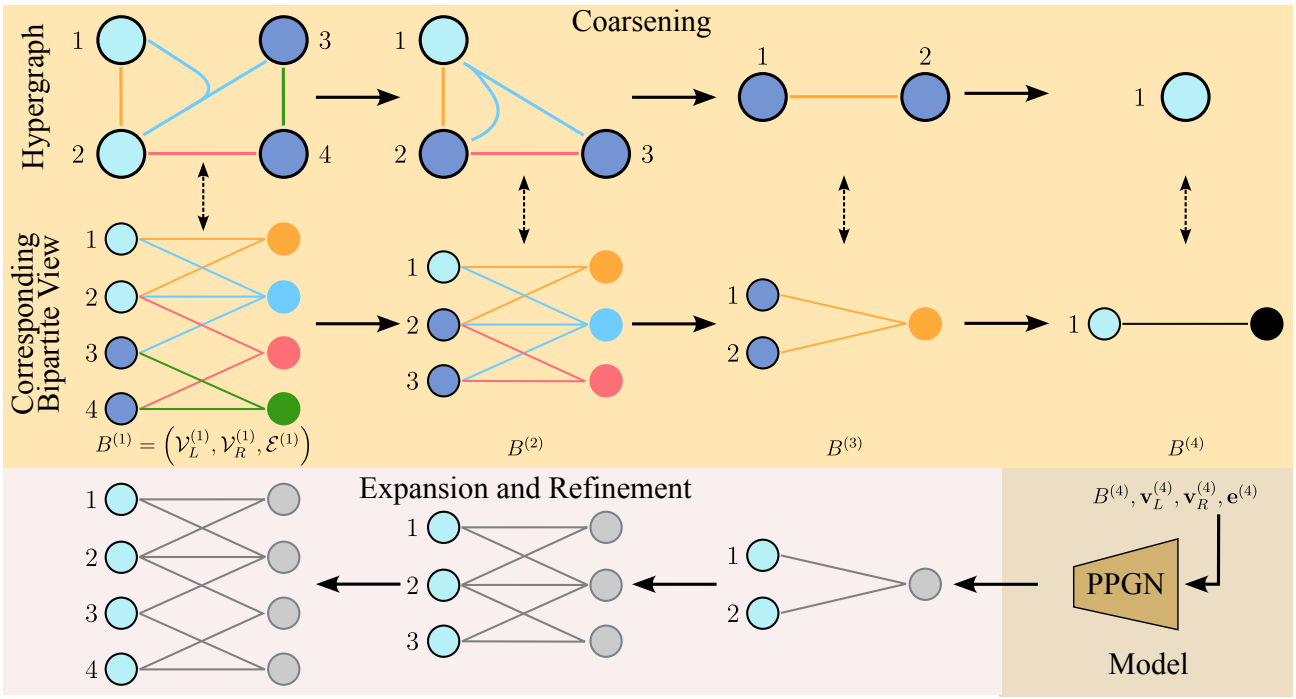


Figure 2: Starting from a hypergraph, our method computes different views of it at increasingly coarser resolutions. An equivalent bipartite representation is maintained in parallel, and a graph neural network model (in our case a PPGN (Maron et al. 2019)) is trained to recover a bipartite representation from its coarser version.

$\{(v, e) \mid v \in \mathcal{V}_L, e \in \mathcal{V}_R, v \in e \text{ in } H\}$. Furthermore, we define the Bolla's Laplacian (Bolla 1993) of a hypergraph as $\mathbf{L}_H = \mathbf{D}_\mathcal{V} - \mathbf{H}\mathbf{D}_\mathcal{E}^{-1}\mathbf{H}^T$, where $\mathbf{D}_\mathcal{V} \in \mathbb{N}^{|\mathcal{V}|\times|\mathcal{V}|}$ is the diagonal degree matrix for the nodes, $\mathbf{D}_\mathcal{E} \in \mathbb{N}^{|\mathcal{E}|\times|\mathcal{E}|}$ is the diagonal matrix of edge orders, and $\mathbf{H} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{E}|}$ is the incidence matrix. The normalized version of $\mathbf{L}_H$, known as Zhou's Laplacian (Zhou, Huang, and Schölkopf 2005), is defined as $\boldsymbol{\mathcal{L}}_H = \mathbf{I} - \mathbf{D}_\mathcal{V}^{-1/2}\mathbf{H}\mathbf{D}_\mathcal{E}^{-1}\mathbf{H}^T\mathbf{D}_\mathcal{V}^{-1/2}$.

The goal of this work is to train a model capable of sampling from the underlying distribution of a given dataset of hypergraphs $(H_1, \ldots, H_N)$, *i.e.*, learning to generate hypergraphs from data. All the proofs of propositions and lemmas of this paper are provided in Appendix A.

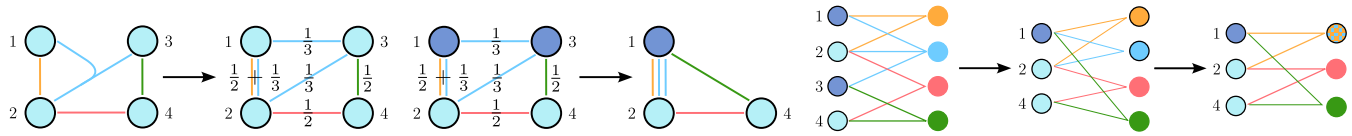
3.2 Overview

The workflow of our approach is illustrated in Figure 2. Our method uses two distinct representations of hypergraphs: the weighted clique expansion and the star expansion. The weighted clique expansion (not depicted in the figure) facilitates the downward traversal of the resolution scales. This representation enables the application of the algorithm proposed by Loukas (2019) to generate reduced versions of the hypergraph while maintaining its spectral properties. Concurrently, we maintain the hypergraph's star expansion, representing it as a bipartite graph. In this representation, one partition corresponds to the hypergraph's nodes (left-side), while the other represents its hyperedges (right-side). Each node is connected to the hyperedges containing it. The bipartite representation proves particularly convenient for ascending the resolution scales. Starting from a coarser representation, the model learns to identify merged nodes and hyperedges, subsequently reconstructing the original bipartite representation. In our implementation, we employ the denoising diffusion model framework (Karras et al. 2022) to model the learning problem: the truth values for the nodes and edges requiring expansion and deletion are noised, and a model is trained to recover the original values. We chose Provably Powerful Graph Network (PPGN) (Maron et al. 2019) as the architecture of this model. Hypergraph generation can then be achieved through an iterative process of increasing resolution, starting from the coarsest bipartite representation, which consists of a pair of connected nodes.

3.3 Descending through the Resolution Scales: Coarsening Sequences

Definition 1 (Graph coarsening)**.** Let $G = (\mathcal{V}, \mathcal{E})$ be an arbitrary graph and $\mathcal{P} = \{\mathcal{V}^{(1)}, \ldots, \mathcal{V}^{(\bar{n})}\}$ be a partitioning[2] of the node set $\mathcal{V}$ such that each set $\mathcal{V}^{(p)} \in \mathcal{P}$ induces a connected subgraph in G. We construct a coarsening $\bar{G}(G, \mathcal{P}) = (\bar{\mathcal{V}}, \bar{\mathcal{E}})$ of G by representing each part $\mathcal{V}^{(p)} \in \mathcal{P}$ as a single node $v^{(p)} \in \bar{\mathcal{V}}$. We add an edge $e_{\{p,q\}} \in \bar{\mathcal{E}}$, between distinct nodes $v^{(p)} \neq v^{(q)} \in \bar{\mathcal{V}}$ in the coarsened graph if and only if there exists an edge $e_{\{i,j\}} \in \mathcal{E}$ between the corresponding disjoint clusters in the original graph, *i.e.*, $v^{(i)} \in \mathcal{V}^{(p)}$ and $v^{(j)} \in \mathcal{V}^{(q)}$.

[2]This implies that $\mathcal{V}^{(p)} \subseteq \mathcal{V}$, $\bigcup_{i=1}^{\bar{n}} \mathcal{V}^{(i)} = \mathcal{V}$, and $\mathcal{V}^{(i)} \cap \mathcal{V}^{(j)} = \emptyset \; \forall \, 1 \leq i, j \leq \bar{n}$.



(a) Weighted clique expansion (b) Coarsening (c) Bipartite representation update

Figure 3: Coarsening: (a) Compute the weighted clique expansion by collapsing each hyperedge into an appropriately weighted clique. (b) Coarsen the clique expansion while preserving the spectral properties of the hypergraph (dark blue nodes). (c) Update the bipartite view: corresponding left-side nodes (in dark blue) are merged, then right-side nodes representing the same hyperedge (circled in black) are merged.

Remark 2. This definition implies partitioning the nodes into different connected sets and merging each part into a cluster. Two clusters are connected if and only if there exists an edge between some node in the first cluster and some node in the second cluster.

We extend Definition 1 to the bipartite representations of hypergraphs:

Definition 3 (Bipartite representation coarsening)**.** Let H be an arbitrary hypergraph, $C = (\mathcal{V}_c, \mathcal{E}_c)$ its weighted clique expansion, $B = (\mathcal{V}_L, \mathcal{V}_R, \mathcal{E})$ its bipartite representation, and $\mathcal{P}_L = \{\mathcal{V}^{(1)}, \dots, \mathcal{V}^{(n)}\}$ a partitioning of the node set $\mathcal{V}_c$ of C (and equivalently of the node set $\mathcal{V}_L$ of B), such that each part $\mathcal{V}^{(p)} \in \mathcal{P}_L$ induces a connected subgraph in C. We construct an intermediate coarsening $\bar{B}(B, \mathcal{P}_L) = (\bar{\mathcal{V}}_L, \mathcal{V}_R, \bar{\mathcal{E}})$ of B by representing each part $\mathcal{V}^{(p)} \in \mathcal{P}_L$ as a single node $v^{(p)} \in \bar{\mathcal{V}}_L$. We add an edge $e_{\{p,q\}} \in \bar{\mathcal{E}}$, between distinct nodes $v^{(p)} \in \bar{\mathcal{V}}_L$ and $v^{(q)} \in \mathcal{V}_R$ in the coarsened graph if and only if there exists an edge $e_{\{i,q\}} \in \mathcal{E}$ between a node $v^{(i)} \in \mathcal{V}^{(p)}$ and right-side node $v^{(q)}$ in the original graph.

Now let $v_1 \sim v_2 \iff \mathcal{N}(v_1) = \mathcal{N}(v_2)$ define an equivalence relation for the right-side nodes in $\mathcal{V}_R$, where $\mathcal{N}(v)$ denotes the set of neighbors of v. This equivalence relation induces a partitioning $\mathcal{P}_R = \{\mathcal{V}_R^{(1)}, \dots, \mathcal{V}_R^{(m)}\}$ of $\mathcal{V}_R$. Finally, we construct the fully coarsened bipartite representation by representing each part $\mathcal{V}^{(p)} \in \mathcal{P}_R$ as a single node $v^{(p)} \in \bar{\mathcal{V}}_R$, in a similar way to $\bar{\mathcal{V}}_L$.

Remark 4. Here, nodes are merged into clusters, and then hyperedges appearing multiple times are merged. This process is illustrated in Figure 3.

We now describe the construction of coarsening sequences for a hypergraph $H = (\mathcal{V}, \mathcal{E})$. This process maintains the weighted clique expansion and the bipartite representation of H in parallel. We leverage the following result:

Proposition 5 (Adapted from Section 6.4 in (Agarwal, Branson, and Belongie 2006) and proved in Appendix A.4)**.** *For an unweighted hypergraph, Bolla's unnormalized Laplacian* $\mathbf{L}_H = \mathbf{D}_\mathcal{V} - \mathbf{H}\mathbf{D}_\mathcal{E}^{-1}\mathbf{H}^T$ *is equal to the unnormalized Laplacian of the associated clique expansion* C*, where each edge* e_{uv} *is weighted by* $\sum_{e \ni u,v;\, e \in \mathcal{E}} \frac{1}{|e|}$.

This proposition establishes that the spectral properties of the hypergraph are equivalent to those of an appropriately weighted clique expansion. Loukas (2019) introduced an algorithm to construct a coarsened version of a graph while preserving a subset of its eigenvalues and eigenvectors. As the weighted clique expansion is a graph, this algorithm can be applied to select groups of nodes for merging. This allows the construction of a coarser view of the hypergraph while preserving relevant spectral properties, which are known to capture important characteristics of the underlying topology. Figure 3a illustrates an example of such a weighted clique expansion.

The coarsening process consists of three steps (illustrated in Figures 3b and 3c):

1. The algorithm by Loukas (2019) operates on the weighted clique expansion to identify a suitable partitioning of nodes, also referred to as "contraction sets".
2. These contraction sets are merged in the weighted clique expansion, and the corresponding left-side nodes in the bipartite representation of the hypergraph are also merged.
3. Finally, the right-side nodes in the bipartite representation of the hypergraph representing the same hyperedge are merged.

This procedure is applied iteratively until we obtain a single-node graph for the weighted clique expansion and a corresponding bipartite graph with a single node on each side for the bipartite representation. It is important to note that in this setting, controlling the merging of hyperedges (right-side nodes for the bipartite representation) is challenging. Empirical experiments have shown that even with few left-node mergings, the right side can easily merge tens of hyperedges into one cluster at once in dense hypergraphs. To avoid this issue, we select the contraction family to be the set of all pairs of adjacent nodes in the clique representation. Therefore, we obtain the following result:

Proposition 6. *For a single merging of two adjacent nodes in the clique representation, at most three hyperedges can be involved in each hyperedge merging in the bipartite representation.*

Remark 7. This proposition holds only for a single merging of a node pair at each coarsening step. In the case of multiple simultaneous mergings, the proposition does not necessarily hold. However, it can be enforced by ensuring that the different contraction sets have disjoint neighborhoods. In our experiments, we instead consider each relevant node merging individually and proceed with it only if every right-side cluster does not exceed three hyperedges. The complete coarsening sampling procedure incorporating this approach is detailed in Algorithm 1 of Appendix E.

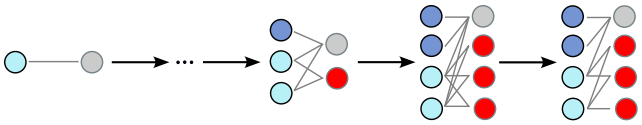


Figure 4: Our model starts from a single pair of linked nodes (in the bipartite representation) and iteratively expands the left-side nodes (in dark blue) and right-side nodes (in red), where each duplicate keeps the connections of its parent node. Then, our method refines the resulting bipartite graph filtering edges to recover an appropriate local structure.

3.4 Ascending through the Resolution Scales: Expansion and Refinement

We now describe the expansion and refinement of the bipartite representation of a hypergraph, which is the inverse of the coarsening process (see Appendix A.3 for a proof). This process operates exclusively on the bipartite representation. At each step, starting from the bipartite representation for a specific resolution level $B = (\mathcal{V}_L, \mathcal{V}_R, \mathcal{E})$, we first select which nodes to duplicate. This selection is encoded via two vectors: $\mathbf{v}_L \in \mathbb{N}^{|\mathcal{V}_L|}$ for left-side nodes, and $\mathbf{v}_R \in \mathbb{N}^{|\mathcal{V}_R|}$ for right-side nodes. These vectors specify the number of times each node needs to be duplicated. Therefore, we expand the bipartite graph by duplicating each node by the specified number. Each duplicate retains the same connectivity as its parent cluster node.

The following definition formalizes this process (illustrated in the two center figures of Figure 4):

Definition 8 (Bipartite graph expansion)**.** Given a bipartite graph $B = (\mathcal{V}_L, \mathcal{V}_R, \mathcal{E})$ and two cluster size vectors $\mathbf{v}_L \in \mathbb{N}^{|\mathcal{V}_L|}$, $\mathbf{v}_R \in \mathbb{N}^{|\mathcal{V}_R|}$, denoting the expansion size of each node, let $\tilde{B}(B, \mathbf{v}_L, \mathbf{v}_R) = (\tilde{\mathcal{V}}_L, \tilde{\mathcal{V}}_R, \tilde{\mathcal{E}})$ denote the expansion of B, whose node sets are given by:

- $\tilde{\mathcal{V}}_L = \mathcal{V}_L^{(1)} \cup \cdots \cup \mathcal{V}_L^{(|\mathcal{V}_L|)}$, where $\mathcal{V}_L^{(p)} = \{v_L^{(p,i)} \mid 1 \leq i \leq \mathbf{v}_L[p]\}$ for $1 \leq p \leq |\mathcal{V}_L|$.
- $\tilde{\mathcal{V}}_R = \mathcal{V}_R^{(1)} \cup \cdots \cup \mathcal{V}_R^{(|\mathcal{V}_R|)}$, where $\mathcal{V}_R^{(p)} = \{v_R^{(p,i)} \mid 1 \leq i \leq \mathbf{v}_R[p]\}$ for $1 \leq p \leq |\mathcal{V}_R|$.

The edge set $\tilde{\mathcal{E}}$ includes all the cluster interconnecting edges: $\{e_{\{p,i;q,j\}} \mid e_{\{p,q\}} \in \mathcal{E}, v_L^{(p,i)} \in \mathcal{V}_L^{(p)}, v_R^{(q,j)} \in \mathcal{V}_R^{(q)}\}$.

After the expansion, we selectively keep or remove edges of the resulting bipartite graph using an edge selection vector $\mathbf{e} \in \{0,1\}^{|\tilde{\mathcal{E}}|}$ (this corresponds to the rightmost step in Figure 4):

Definition 9 (Bipartite representation refinement)**.** Given a bipartite graph $\tilde{B} = (\tilde{\mathcal{V}}_L, \tilde{\mathcal{V}}_R, \tilde{\mathcal{E}})$ and an edge selection vector $\mathbf{e} \in \{0,1\}^{|\tilde{\mathcal{E}}|}$, let $B(\tilde{B}, \mathbf{e}) = (\mathcal{V}_L, \mathcal{V}_R, \mathcal{E})$ denote the refinement of $\tilde{B}$, where: $\mathcal{V}_L = \tilde{\mathcal{V}}_L$, $\mathcal{V}_R = \tilde{\mathcal{V}}_R$, $\mathcal{E} \subseteq \tilde{\mathcal{E}}$ such that the i-th edge $e_{(i)} \in \mathcal{E}$ if and only if $\mathbf{e}[i] = 1$.

The process of generating a hypergraph consists of the following steps:

1. Start from a bipartite graph containing only two nodes linked by an edge: $B^{(L)} = (\{1\}, \{2\}, \{(1,2)\})$ (leftmost figure of Figure 4).
2. Iteratively expand and refine the current bipartite representation to add details until the desired size is attained:
$$B^{(l)} \xrightarrow{\text{expand}} \tilde{B}^{(l-1)} \xrightarrow{\text{refine}} B^{(l-1)}$$
3. Once the final bipartite representation is generated, recover the associated hypergraph by collapsing each node on the right-side into a hyperedge.

Remark 10. Our generation part differs from that by Bergmeister et al. (2024) in the expansion step. While they initially retain all possible edges between connected clusters before selection, we treat hyperedges analogously to nodes. This constraint is necessary to maintain computational feasibility due to the exponential growth of potential hyperedges in hypergraphs.

3.5 Probabilistic Modeling

We now formalize our learning problem, generalizing the approach by Bergmeister et al. (2024). Given a dataset $\{H^{(1)}, \ldots, H^{(N)}\}$ of i.i.d. hypergraph samples, our goal is to fit a distribution $p(H)$ that closely approximates the unknown true generative process. We model the marginal likelihood of a hypergraph H as the sum of likelihoods over expansion sequences of its bipartite representation B:

$$p(H) = p(B) = \sum_{\varpi \in \Pi(B)} p(\varpi), \tag{1}$$

where $\Pi(B)$ denotes the set of all possible expansion sequences $(B^{(L)} = (\{1\}, \{2\}, \{(1,2)\}), B^{(1)}, \ldots, B^{(0)} = B)$ from a minimal bipartite graph to the target hypergraph's bipartite representation B. Each $B^{(l-1)}$ is a refined expansion of its predecessor, as per Definitions 8 and 9.

Assuming a Markovian structure, we factorize the likelihood as:

$$p(\varpi) = \underbrace{p(B^{(L)})}_{1} \cdot \prod_{l=L}^{1} p(B^{(l-1)} | B^{(l)}) \tag{2}$$

$$= \prod_{l=L}^{1} p(\mathbf{e}^{(l-1)} | \tilde{B}^{(l-1)}) p(\mathbf{v}_L^{(l)}, \mathbf{v}_R^{(l)} | B^{(l)}). \tag{3}$$

To avoid modeling two separate distributions $p(\mathbf{e}^{(l)} | \tilde{B}^{(l)})$ and $p(\mathbf{v}_L^{(l)}, \mathbf{v}_R^{(l)} | B^{(l)})$, we rearrange terms as:

$$p(\varpi) = p(\mathbf{e}^{(0)} | \tilde{B}^{(0)}) \cdot \underbrace{p(\mathbf{v}_L^{(L)}, \mathbf{v}_R^{(L)} | B^{(L)})}_{p(\mathbf{v}_L^{(L)}, \mathbf{v}_R^{(L)})} \cdot \left[\prod_{l=L-1}^{1} p(\mathbf{v}_L^{(l)}, \mathbf{v}_R^{(l)} | B^{(l)}) p(\mathbf{e}^{(l)} | \tilde{B}^{(l)}) \right]. \tag{4}$$

We model $\mathbf{v}_L^{(l)}$ and $\mathbf{v}_R^{(l)}$ to be conditionally independent of $\tilde{B}^{(l)}$ given $B^{(l)}$, *i.e.*, $p(\mathbf{v}_L^{(l)}, \mathbf{v}_R^{(l)} | B^{(l)}, \tilde{B}^{(l)}) = p(\mathbf{v}_L^{(l)}, \mathbf{v}_R^{(l)} | B^{(l)})$. This allows us to finally write:

$$p(\mathbf{v}_L^{(l)}, \mathbf{v}_R^{(l)} | B^{(l)}) p(\mathbf{e}^{(l)} | \tilde{B}^{(l)}) = p(\mathbf{v}_L^{(l)}, \mathbf{v}_R^{(l)}, \mathbf{e}^{(l)} | \tilde{B}^{(l)}). \tag{5}$$

Model	SBM Hypergraphs ($n_{avg} = 31.73$, $std = 0.55$)					Ego Hypergraphs ($n_{avg} = 109.71$, $std = 10.23$)					Tree Hypergraphs ($n_{avg} = 32$, $std = 0$)				
	Valid SBM ↑	**Node Num** ↓	**Node Deg** ↓	**Edge Size** ↓	**Spectral** ↓	**Valid Ego** ↑	**Node Num** ↓	**Node Deg** ↓	**Edge Size** ↓	**Spectral** ↓	**Valid Tree** ↑	**Node Num** ↓	**Node Deg** ↓	**Edge Size** ↓	**Spectral** ↓
HyperPA	2.5%	**0.075**	4.062	0.407	0.273	0%	35.83	2.590	0.423	0.237	0%	2.350	0.315	0.284	0.159
VAE	0%	0.375	1.280	1.059	0.024	0%	47.58	0.803	1.458	0.133	0%	9.700	0.072	0.480	0.124
GAN	0%	1.200	2.106	1.203	0.059	0%	60.35	0.917	1.665	0.230	0%	6.000	0.151	0.469	0.089
Diffusion	0%	0.150	1.717	1.390	0.031	0%	**4.475**	3.984	2.985	0.190	0%	2.225	1.718	1.922	0.127
HYGENE	**65%**	0.525	**0.321**	**0.002**	**0.010**	**90%**	12.55	**0.063**	**0.220**	**0.004**	**77.5%**	**0.000**	**0.059**	**0.108**	**0.012**

Table 1: Comparison between HYGENE and other baselines for the SBM, Ego, and Tree hypergraphs.

3.6 Implementation

We employ the EDM denoising diffusion framework (Karras et al. 2022) to model the probability $p(\mathbf{v}_L, \mathbf{v}_R, \mathbf{e}|B)$: $\mathbf{v}_L$, $\mathbf{v}_R$, and $\mathbf{e}$ are treated as node and edge features of the bipartite representation; these features are noised and a model is trained to recover the original features. Our model consists of an embedding layer for each vector, followed by a PPGN (Maron et al. 2019) feeding into one final output layer. See Appendix C.1 for details of the architecture.

The main implementation details are outlined below, with further details provided in Appendix C.2.

1. Similarly to (Bergmeister et al. 2024), we use a deterministic expansion size procedure where only a predefined number of nodes are expanded at each iteration, those being the most probable according to the model, while the others are left unchanged.
2. We also reuse the perturbed expansion, where the bipartite graph is noised through the introduction of random edges connecting a node to others located within a predefined radius around it.
3. We extend spectral conditioning to hypergraphs, wherein spectral properties of the target hypergraph are used as conditioning during the prediction of $B^{(l)}$.

Indeed, the spectrum of $B^{(l+1)}$ approximates the target spectrum due to the spectral preservation during coarsening: using Lemma 1 by Agarwal, Branson, and Belongie (2006), we can easily prove (see Appendix A.4):

$$Sp(\mathcal{L}_B) = \left\{1 \pm \sqrt{1-\lambda} \mid \lambda \in Sp(\mathcal{L}_H)\right\} \subset [0, 2], \quad (6)$$

where $Sp(\mathcal{L}_B)$ and $Sp(\mathcal{L}_H)$ are the spectra of the *normalized* Laplacians of the bipartite representation and the hypergraph, respectively. There is also equivalence for the eigenspaces (see the proof in Appendix A.4). Consequently, preserving the k smallest non-zero eigenvalues of the *unnormalized* Laplacian of the weighted clique expansion also preserves the k smallest non-zero (and k largest non-equal to 2) eigenvalues of the normalized Laplacian of the bipartite representation.

During hypergraph reconstruction from its bipartite representation, isolated nodes and empty hyperedges (corresponding to empty rows and columns in the incidence matrix) are discarded.

4 Experiments and Results

In this section, we detail our experimental setup, covering datasets and evaluation metrics. Then, we compare our approach against the following baselines: HyperPA (Do et al. 2020), a Variational Autoencoder (VAE) (Kingma and Welling 2013), a Generative Adversarial Network (GAN) (Goodfellow et al. 2020), and a standard 2D diffusion model (Ho, Jain, and Abbeel 2020) trained on incidence matrix images, where hyperedge membership is represented by white pixels and absence by black pixels. Finally, we ablate on the spectrum-preserving coarsening and the upper bound for the number of hyperedges defined in Proposition 6.

Our goal is threefold: (i) proving that HYGENE can generate the desired hyperedge distribution, (ii) proving that HYGENE can closely mimic a range of strict structural properties, and (iii) proving the importance of our components, *i.e.*, spectrum-preserving coarsening and upper bounding the size of hyperedge clusters during coarsening. Detailed numerical results are available in Appendix G, and Appendix H provides several visualizations of our generated hypergraphs.

Datasets. We evaluate our method on four synthetic hypergraph datasets: Erdős–Rényi (ER) (Erdős and Rényi 1960), Stochastic Block Model (SBM) (Kim, Bandeira, and Goemans 2018), Ego (Comrie and Kleinberg 2021), and Tree (Nieminen and Peltola 1999). Furthermore, we also test HYGENE on topologies of low-poly featureless versions of three classes of ModelNet40 (Wu et al. 2015) converted to hypergraphs: *plant*, *piano*, and *bookshelf*. Each dataset is split into 128 training, 32 validation, and 40 test hypergraphs. More details can be found in Appendix D.1.

Metrics. Our metrics measure: (i) overall structural similarities like *Node Num* (difference in the number of nodes), *Node Deg* (difference in node degrees), and *Edge Size* (difference in the size of the hyperedges); (ii) topological properties by computing the average difference of the *Spectral* properties. For datasets with specific structural requirements, *Valid* metrics assess the fraction of generated samples satisfying these properties. Lower values indicate better performance for all metrics except *Valid*, where higher is better. More details can be found in Appendix D.2.

Comparison with the Baselines. Tables 1 and 2 show the comparison of HYGENE against our baselines. We observe that the proposed method effectively captures the edge size distribution and successfully imitates structural properties

Model	Erdos-Renyi Hypergraphs ($n_{avg} = 32$, $std = 0.07$)				ModelNet40 Piano ($n_{avg} = 177.29$, $std = 57.11$)				ModelNet40 Plant ($n_{avg} = 124.86$, $std = 87.88$)				ModelNet40 Bookshelf ($n_{avg} = 119.38$, $std = 68.20$)			
	Node Num ↓	**Node Deg** ↓	**Edge Size** ↓	**Spectral** ↓	**Node Num** ↓	**Node Deg** ↓	**Edge Size** ↓	**Spectral** ↓	**Node Num** ↓	**Node Deg** ↓	**Edge Size** ↓	**Spectral** ↓	**Node Num** ↓	**Node Deg** ↓	**Edge Size** ↓	**Spectral** ↓
HyperPA	**0.000**	5.530	0.183	0.177	0.825	9.254	**0.023**	**0.067**	10.83	6.566	0.046	0.061	8.025	7.562	0.044	**0.048**
VAE	0.100	2.140	0.540	0.035	75.35	8.060	1.686	0.396	76.15	3.895	1.573	0.205	47.45	6.190	1.520	0.190
GAN	0.675	2.560	0.657	0.048	**0.000**	409.0	86.38	0.697	**0.000**	378.1	56.35	0.364	**0.000**	397.2	46.30	0.476
Diffusion	0.050	2.225	0.781	0.014	0.050	20.90	4.192	0.113	0.025	21.03	3.439	0.069	**0.000**	20.36	2.346	0.079
HYGENE	0.775	**0.475**	**0.012**	**0.006**	42.52	**6.290**	0.027	0.117	68.38	**2.428**	**0.027**	**0.034**	69.73	**1.050**	**0.034**	0.068

Table 2: Comparison between HYGENE and other baselines for the ER and ModelNet40 hypergraphs.

Upper Bound	Spec. Pr. Coarsen.	SBM Hypergraphs				Ego Hypergraphs				Tree Hypergraphs				Erdős–Rényi Hypergraphs		
		Valid SBM ↑	**Node Deg** ↓	**Edge Size** ↓	**Spectral** ↓	**Valid Ego** ↑	**Node Deg** ↓	**Edge Size** ↓	**Spectral** ↓	**Valid Tree** ↑	**Node Deg** ↓	**Edge Size** ↓	**Spectral** ↓	**Node Deg** ↓	**Edge Size** ↓	**Spectral** ↓
✗	✓	57.5%	1.234	0.006	**0.009**	77.5%	0.861	0.654	0.149	20%	0.134	0.088	0.014	1.018	0.045	0.009
✓	✗	47.5%	**0.148**	0.005	0.011	77.5%	0.115	**0.146**	**0.004**	**77.5%**	0.072	**0.015**	0.026	1.015	0.124	0.014
✓	✓	**65%**	0.321	**0.002**	0.010	**90%**	**0.063**	0.220	**0.004**	**77.5%**	**0.059**	0.108	**0.012**	**0.475**	**0.012**	**0.006**

Table 3: Ablation studies on the upper bound in the number of hyperedges and the spectrum-preserving coarsening.

across datasets. This is particularly evident in the Ego dataset, where our approach uniquely generates correct ego hypergraphs with a high success rate of 90%. In some cases, HyperPA has a similar or better performance compared to HYGENE, but this baseline requires to know a-priori the true node degree and hyperedge distributions on a per-hypergraph basis, whereas our model only requires the desired number of nodes.

The primary advantage of HYGENE over other baseline approaches lies in its comprehension of hypergraph structure. This is particularly evident in the *Valid* metrics, where only HYGENE achieves satisfactory results. Indeed, node degrees and hyperedge sizes can be replicated by merely outputting the correct density of white pixels on the incidence matrix images. This explains the satisfactory results sometimes achieved by these baselines for *Node Deg* and *Edge Size*. However, the underlying structure cannot be captured this way, and our baselines fail the *Valid* metrics.

While one might consider using graph generators to produce hypergraph representations, this approach is infeasible because: (i) generating the clique expansion and recovering the associated hypergraph is an NP-hard problem as it requires the enumeration of all cliques; and (ii) for the bipartite representation, determining which side corresponds to nodes and which to hyperedges is non-trivial. Additionally, we empirically observe that graph-based models often struggle to generate valid bipartite graphs. For example, we only obtained 30% of correct bipartite graphs for both models by Bergmeister et al. (2024) and Vignac et al. (2023).

Ablation Studies. Table 3 shows the results of our ablation studies. First, we observe that not enforcing an upper limit on the hyperedge cluster sizes makes the hyperedge generation task more difficult: for the four datasets, *Node Deg* greatly increases. This is especially harmful to the *Valid Tree* and the Ego's *Spectral* metrics as their structure requires very specific sparsity properties. For the four datasets, the model overestimates the correct number of hyperedges and produces denser hypergraphs.

Then, we also observe that the effects of spectrum-preserving coarsening are more subtle. SBM and Ego hypergraphs suffer the most from its absence. This is expected since failing to preserve their structure during coarsening reduces the available information the model possesses during generation. Tree hypergraphs are not heavily impacted due to their relative absence of global structure, whereas ER hypergraphs suffer in the *Node Deg* and *Edge Size* metrics as the model struggles to correctly generate the hyperedges. This is also expected since the spectrum of the hypergraph contains information on the hyperedge distribution.

Limitations. The mesh datasets reveal that HYGENE faces difficulties in accurately producing the specified number of nodes and hyperedges. The model appears to sample from the underlying distribution of hypergraph sizes rather than adhering to the node count directive provided during the generation process. Notably, the *Node Num* metric approximates the standard deviation (std) of node counts for each dataset. We hypothesize it stems from an inability to correctly estimate the number of hyperedges. The model then discards the excess nodes by disconnecting them to keep the targeted properties in the remainder of the hypergraph.

5 Conclusion

In this work, we introduced HYGENE which is, to the best of our knowledge, the first attempt at diffusion-based hypergraph generation. We generalized the iterative local expansion scheme by Bergmeister et al. (2024) and the coarsening process by Loukas (2019) for hypergraphs. Therefore, we introduced an iterative local expansion procedure for the generation of hyperedges. We trained a denoising diffusion model and successfully tested the ability of our method to generate hypergraphs sampled from specific distributions. This work provides a key contribution to graph generation, being one of the first able to directly generate hypergraphs.

Acknowledgments

The authors acknowledge the ANR – FRANCE (French National Research Agency) for its financial support of the System On Chip Design leveraging Artificial Intelligence (SODA) project under grant ANR-23-IAS3-0004 and the JCJC project DeSNAP ANR-24-CE23-1895-01. This project has also been partially funded by the Hi!PARIS Center on Data Analytics and Artificial Intelligence.

References

Agarwal, S.; Branson, K.; and Belongie, S. 2006. Higher order learning with graphs. In *International Conference on Machine Learning*.

Aksoy, S. G.; Joslyn, C.; Marrero, C. O.; Praggastis, B.; and Purvine, E. 2020. Hypernetwork science via high-order hypergraph walks. *EPJ Data Science*.

Arafat, N. A.; Basu, D.; Decreusefond, L.; and Bressan, S. 2020. Construction and random generation of hypergraphs with prescribed degree and dimension sequences. In *International Conference on Database and Expert Systems Applications*.

Bergmeister, A.; Martinkus, K.; Perraudin, N.; and Wattenhofer, R. 2024. Efficient and scalable graph generation through iterative local expansion. In *International Conference on Learning Representations*.

Bolla, M. 1993. Spectra, Euclidean representations and clusterings of hypergraphs. *Discrete Mathematics*.

Chen, X.; He, J.; Han, X.; and Liu, L.-P. 2023. Efficient and degree-guided graph generation via discrete diffusion modeling. In *International Conference on Machine Learning*.

Comrie, C.; and Kleinberg, J. 2021. Hypergraph ego-networks and their temporal evolution. In *IEEE International Conference on Data Mining*.

Do, M. T.; Yoon, S.-e.; Hooi, B.; and Shin, K. 2020. Structural patterns and generative models of real-world hypergraphs. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Dupagne, A.; and Teller, A. 1998. Hypergraph formalism for urban form specification. In *COST C4 Final Conference, Kiruna*.

Erdős, P.; and Rényi, A. 1960. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci*.

Estrada, E.; and Rodríguez-Velázquez, J. A. 2006. Subgraph centrality and clustering in complex hyper-networks. *Physica A: Statistical Mechanics and its Applications*.

Gong, X.; Higham, D. J.; and Zygalakis, K. 2023. Generative hypergraph models and spectral embedding. *Scientific Reports*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2020. Generative adversarial networks. *Communications of the ACM*.

Grzesiak-Kopeć, K.; Oramus, P.; and Ogorzałek, M. 2017. Hypergraphs and extremal optimization in 3D integrated circuit design automation. *Advanced Engineering Informatics*.

Guo, Y.; Zou, D.; and Lerman, G. 2023. An Unpooling Layer for Graph Generation. In *International Conference on Artificial Intelligence and Statistics*.

Higham, D. J.; and De Kergorlay, H.-L. 2021. Epidemics on hypergraphs: Spectral thresholds for extinction. *Proceedings of the Royal Society A*.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*.

Kajino, H. 2019. Molecular hypergraph grammar with its application to molecular optimization. In *International Conference on Machine Learning*.

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

Kim, C.; Bandeira, A. S.; and Goemans, M. X. 2018. Stochastic block model for hypergraphs: Statistical limits and a semidefinite programming approach. *arXiv preprint arXiv:1807.02884*.

Kingma, D. P.; and Welling, M. 2013. Auto-encoding variational Bayes. In *International Conference on Learning Representations*.

Kong, L.; Cui, J.; Sun, H.; Zhuang, Y.; Prakash, B. A.; and Zhang, C. 2023. Autoregressive diffusion model for graph generation. In *International Conference on Machine Learning*.

Lim, D.; Robinson, J.; Zhao, L.; Smidt, T.; Sra, S.; Maron, H.; and Jegelka, S. 2022. Sign and basis invariant networks for spectral graph representation learning. In *International Conference on Learning Representations*.

Lin, Z.; Yan, Q.; Liu, W.; Wang, S.; Wang, M.; Tan, Y.; and Yang, C. 2023. Automatic hypergraph generation for enhancing recommendation with sparse optimization. *IEEE Transactions on Multimedia*.

Loukas, A. 2019. Graph reduction with spectral and cut guarantees. *Journal of Machine Learning Research*.

Luo, Z.; Hy, T. S.; Tabaghi, P.; Defferrard, M.; Rezaei, E.; Carey, R. M.; Davis, R.; Jain, R.; and Wang, Y. 2024. DE-HNN: An effective neural model for circuit netlist representation. In *International Conference on Artificial Intelligence and Statistics*.

Maron, H.; Ben-Hamu, H.; Serviansky, H.; and Lipman, Y. 2019. Provably powerful graph networks. In *Advances in Neural Information Processing Systems*.

Martinkus, K.; Loukas, A.; Perraudin, N.; and Wattenhofer, R. 2022. Spectre: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *International Conference on Machine Learning*.

Murgas, K. A.; Saucan, E.; and Sandhu, R. 2022. Hypergraph geometry reflects higher-order dynamics in protein interaction networks. *Scientific Reports*.

Nieminen, J.; and Peltola, M. 1999. Hypertrees. *Applied mathematics letters*.

Niu, C.; Song, Y.; Song, J.; Zhao, S.; Grover, A.; and Ermon, S. 2020. Permutation invariant graph generation via score-based generative modeling. In *International Conference on Artificial Intelligence and Statistics*.

Rahman, A.; Poirel, C. L.; Badger, D. J.; and Murali, T. 2012. Reverse engineering molecular hypergraphs. In *ACM Conference on Bioinformatics, Computational Biology and Biomedicine*.

Simonovsky, M.; and Komodakis, N. 2018. GraphVAE: Towards generation of small graphs using variational autoencoders. In *International Conference on Artificial Neural Networks*.

Starostin, N.; and Balashov, V. 2008. The use of hypergraphs for solving the problem of orthogonal routing of large-scale integrated circuits with an irregular structure. *Journal of Communications Technology and Electronics*.

Vignac, C.; Krawczuk, I.; Siraudin, A.; Wang, B.; Cevher, V.; and Frossard, P. 2023. Digress: Discrete denoising diffusion for graph generation. In *International Conference on Learning Representations*.

Vishnoi, N. K. 2013. Lx = b. *Foundations and Trends® in Theoretical Computer Science*, 8(1–2): 1–141.

Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; and Xiao, J. 2015. 3D shapenets: A deep representation for volumetric shapes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

You, J.; Ying, R.; Ren, X.; Hamilton, W.; and Leskovec, J. 2018. GraphRNN: Generating realistic graphs with deep auto-regressive models. In *International Conference on Machine Learning*.

Zhou, D.; Huang, J.; and Schölkopf, B. 2005. Beyond pairwise classification and clustering using hypergraphs. Technical Report 143, Max Planck Institute for Biological Cybernetics.

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

Zhu, Y.; Ouyang, Z.; Liao, B.; Wu, J.; Wu, Y.; Hsieh, C.-Y.; Hou, T.; and Wu, J. 2023. MOLHF: A hierarchical normalizing flow for molecular graph generation. In *International Joint Conference on Artificial Intelligence*.

A Proofs

A.1 Laplacians equality

Proposition (Adapted from Section 6.4 of Agarwal, Branson, and Belongie (2006))**.** *For an unweighted hypergraph, Bolla's unnormalized Laplacian $\mathbf{L}_H = \mathbf{D}_\mathcal{V} - \mathbf{H}\mathbf{D}_\mathcal{E}^{-1}\mathbf{H}^T$ is equal to the unnormalized Laplacian of the associated clique expansion C, where each edge e_{uv} is weighted by $\sum_{e \ni u,v;\, e\in\mathcal{E}} \frac{1}{|e|}$.*

Proof. Let $H = (\mathcal{V}, \mathcal{E})$ be a hypergraph and C its weighted clique expansion. The unnormalized Laplacian of H is given by $\mathbf{L}_H = \mathbf{D}_\mathcal{V} - \mathbf{H}\mathbf{D}_\mathcal{E}^{-1}\mathbf{H}^T$.

For indices $i \neq j$,

$$(\mathbf{L}_H)_{ij} = -\sum_{e\ni i,j; e\in\mathcal{E}} \frac{1}{|e|} = -\mathbf{A}_{ij} \tag{7}$$

where $\mathbf{A}$ is the adjacency matrix of the weighted clique expansion. For an index i on the diagonal, denoting d_i the number of hyperedges containing node i,

$$(\mathbf{L}_H)_{ii} = d_i - \sum_{e\ni i; e\in\mathcal{E}} \sum_{j\in e; j\neq i} \frac{1}{|e|} = d_i - \sum_{e\ni i; e\in\mathcal{E}} \frac{|e|-1}{|e|} = \sum_{e\ni i; e\in\mathcal{E}} \left(1 - \frac{|e|-1}{|e|}\right) = \sum_{e\ni i; e\in\mathcal{E}} \frac{1}{|e|} = \mathbf{D}_{ii} \tag{8}$$

where $\mathbf{D}$ is the degree matrix of the weighted clique expansion. Thus $\mathbf{L}_H = \mathbf{L}_C$. □

A.2 Upper Bound on Hyperedge Cluster Size

Proposition. *In a single merging of two adjacent nodes in the clique representation, at most three hyperedges can be involved in each hyperedge merging in the bipartite representation.*

Proof. We begin by establishing the following lemma:

Lemma. *For each node merging, a hyperedge can merge with at most one other hyperedge of the same size.*

Let n_1 and n_2 be the nodes being merged. Begin by this observation: consider two merging hyperedges $e_1 = (e_{11}, \ldots, e_{1k})$ and $e_2 = (e_{21}, \ldots, e_{2l})$ (not necessarily of the same size). They must be identical after merging. If both contain n_1 and n_2, this implies that they are of the same size k and that their other $k-2$ vertices must be the same, as they remain unchanged. Thus, $e_1 = e_2$.

Now, we prove the lemma by contradiction. Assume there exist three distinct hyperedges of the same size $e_1 = (e_{11}, \ldots, e_{1k})$, $e_2 = (e_{21}, \ldots, e_{2k})$, and $e_3 = (e_{31}, \ldots, e_{3k})$ that merge into a cluster when nodes n_1 and n_2 merge. Only one of n_1 or n_2 can appear in each hyperedge (otherwise, they would be of different sizes after merging or would violate our initial observation).

For these hyperedges to merge, their $k-1$ other vertices must be identical. With only two possible choices (n_1 or n_2) for three hyperedges, at least two must be identical, contradicting our assumption of three distinct hyperedges.

For the main result, we again use contradiction. Assume four hyperedges e_1, e_2, e_3, and e_4 merge into a hyperedge cluster when nodes n_1 and n_2 merge. By our initial observation, either one hyperedge is of order k and contains both n_1 and n_2 while the others are of order $k-1$ with only one of the two nodes, or all are of the same order with only one node appearing in each hyperedge.

In either case, at least three hyperedges of the same order must merge, contradicting our lemma. □

A.3 Inverting Coarsening through Expansion and Refinement

Here we demonstrate that each coarsened bipartite graph (as per Definition 1) can be inverted using specific expansion and refinement steps. Our proof adapts the approach outlined in Appendix A of (Bergmeister et al. 2024).

Let H be an arbitrary hypergraph, C its clique representation, and $B = (\mathcal{V}_L, \mathcal{V}_R, \mathcal{E})$ its bipartite representation. Let $\mathcal{P} = (\mathcal{P}_L, \mathcal{P}_R)$ be a partitioning of $\mathcal{V}_L$ and $\mathcal{V}_R$ according to Definition 3 based on C. Furthermore, let $B^c = (\mathcal{V}_L^c, \mathcal{V}_R^c, \mathcal{E}^c) = \bar{B}(B, \mathcal{P})$ denote the coarsened bipartite representation as per Definition 3. We will now construct the expansion and refinement vectors that recover the original bipartite graph B from its coarsening B^c.

We begin with the expansion by defining vectors $\mathbf{v}_L \in \mathbb{N}^{|\mathcal{P}_L|}$ and $\mathbf{v}_R \in \mathbb{N}^{|\mathcal{P}_R|}$ as follows:

$$\begin{aligned} \mathbf{v}_L[p] &= |\mathcal{V}_L^{(p)}| \text{ for all } \mathcal{V}_L^{(p)} \in \mathcal{P}_L \\ \mathbf{v}_R[p] &= |\mathcal{V}_R^{(p)}| \text{ for all } \mathcal{V}_R^{(p)} \in \mathcal{P}_R \end{aligned} \tag{9}$$

Let $B^e = (\mathcal{V}_L^e, \mathcal{V}_R^e, \mathcal{E}^e) = \tilde{B}(B^c, \mathbf{v}_L, \mathbf{v}_R)$ represent the expanded graph as defined in Definition 8. Notably, the node sets of B and B^e have the same cardinality. Thus, we can establish two bijections $\phi_L : \mathcal{V}_L \to \mathcal{V}_L^e$ and $\phi_R : \mathcal{V}_R \to \mathcal{V}_R^e$ between them.

Here, the i-th node in the p-th part of $\mathcal{P}_L$ maps to the corresponding node $v_L^{(p,i)} \in \mathcal{V}_L^e$ in the expanded graph for ϕ_L, with a similar mapping for ϕ_R.

This construction ensures that the edge set of B^e is a superset of the original graph B's edge set. To illustrate, consider an arbitrary edge $e_{\{i,j\}} \in \mathcal{E}$. Due to the bipartite nature of the graphs, $v_L^{(i)}$ and $v_R^{(j)}$ must lie in different partitions $\mathcal{V}_L^{(p)}$ and $\mathcal{V}_R^{(q)}$ respectively. An edge in the original representation implies that the parts representing nodes $v_L^{(p)} \in \mathcal{V}_L^c$ and $v_R^{(q)} \in \mathcal{V}_R^c$ are connected in B^c. Consequently, when expanding $v_L^{(p)}$ and $v_R^{(q)}$, all $|\mathcal{V}_L^{(p)}|$ nodes associated with $v_L^{(p)}$ connect to all $|\mathcal{V}_R^{(q)}|$ nodes associated with $v_R^{(q)}$ in B^e. Specifically, $v_L^{(\phi_L(i))}$ and $v_R^{(\phi_R(j))}$ are connected in B^e.

For the refinement step, we define the vector $\mathbf{e} \in \{0,1\}^{|\mathcal{E}^e|}$ as follows: given an arbitrary ordering of edges in $\mathcal{E}^e$, let $\mathbf{e}_{(i)} \in \mathcal{E}^e$ denote the i-th edge in this ordering. We set:

$$\mathbf{e}[i] = \begin{cases} 1 & \text{if } e^{\{\phi_L^{-1}(i), \phi_R^{-1}(j)\}} \in \mathcal{E} \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

As per Definition 9, the refined graph is then given by $B^r = \bar{B}(B^e, \mathbf{e}) = \bar{B}(\tilde{B}(B^c, \mathbf{v}_L, \mathbf{v}_R), \mathbf{e})$, which is isomorphic to the original bipartite graph B.

A.4 Spectral Equivalence of Hypergraphs and Their Bipartite Representations

Proposition 11. *Let H be a hypergraph and B its bipartite representation. Denote $\mathcal{L}_H$ and $\mathcal{L}_B$ their respective normalized Laplacian matrix. We have the following equality:*

$$Sp(\mathcal{L}_B) = \left\{1 \pm \sqrt{1-\lambda} \;\middle|\; \lambda \in Sp(\mathcal{L}_H)\right\} \subset [0,2] \tag{11}$$

Proof. Let $H = (\mathcal{V}, \mathcal{E})$ be a hypergraph, and $B = (\mathcal{V}_L, \mathcal{V}_\mathcal{R}, \mathcal{E}_B)$ its bipartite representation, with $\mathcal{V}_B = \mathcal{V}_L \cup \mathcal{V}_R$. Following (Agarwal, Branson, and Belongie 2006), we can express the *normalized* graph Laplacian of B as:

$$\mathcal{L}_B = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} = \begin{bmatrix} \mathbf{I} & -\mathbf{D}_v^{-\frac{1}{2}}\mathbf{H}\mathbf{D}_e^{-\frac{1}{2}} \\ -\mathbf{D}_e^{-\frac{1}{2}}\mathbf{H}^\top\mathbf{D}_v^{-\frac{1}{2}} & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & -\mathbf{M} \\ -\mathbf{M}^T & \mathbf{I} \end{bmatrix} \tag{12}$$

where $\mathbf{M} = \mathbf{D}_v^{-\frac{1}{2}}\mathbf{H}\mathbf{D}_e^{-\frac{1}{2}}$. The hypergraph *normalized* Laplacian is then given by $\mathcal{L}_H = \mathbf{I} - \mathbf{M}\mathbf{M}^T$.

$\subseteq$**:** Let μ be an eigenvalue of $\mathcal{L}_B$ with eigenvector $\mathbf{v} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}$. Then $\mathcal{L}_B\mathbf{v} = \mu\mathbf{v}$. We find that $\mathbf{y} = 0$ if $\mu = 0$, and $\mathbf{y} = \frac{\mathbf{M}^T\mathbf{x}}{1-\mu}$ otherwise, implying $\mathbf{v}$ is linear in $\mathbf{x}$. We also have $\mathbf{M}\mathbf{M}^T\mathbf{x} = (1-\mu)^2\mathbf{x}$, so $1-(1-\mu)^2$ is an eigenvalue of $\mathcal{L}_H$ with the same multiplicity as in $\mathcal{L}_B$, *i.e.*, all eigenvalue μ of $\mathcal{L}_B$ can be written as $1 \pm \sqrt{1-\lambda}$ with λ eigenvalue of $\mathcal{L}_H$.

$\supseteq$**:** Let λ be an eigenvalue of $\mathcal{L}_H$ with eigenvector $\mathbf{x}$. Define $\mu = 1 \pm \sqrt{1-\lambda}$ and $\mathbf{v} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \end{pmatrix}$. If $\lambda = 0$, set $\mathbf{y} = 0$; otherwise, set $\mathbf{y} = \frac{\mathbf{M}^T\mathbf{x}}{1-\mu}$. In both cases, $\mathcal{L}_B\mathbf{v} = \mu\mathbf{v}$, showing that μ is an eigenvalue of $\mathcal{L}_B$ with the same multiplicity as λ in $\mathcal{L}_H$. □

B Complexity Analysis

Here we analyze the asymptotic complexity of our algorithm, generalizing the approach by (Bergmeister et al. 2024). To generate a hypergraph of n nodes, m hyperedges and k incidences, our algorithm computes an expansion sequence of bipartite graphs $(B^{(L)} = (\{1\}, \{2\}, \{(1,2)\}), B^{(L-1)}, \ldots, B^{(0)} = B)$ where the last one is equal to the bipartite representation of the generated hypergraph. In the following, n, m and k will equivalently denote the number of nodes and number of left-side nodes, number of hyperedges and number of right-side nodes, and number of incidences and number of edges of respectively the hypergraph and its associated bipartite representation.

First, in our setting there exists $\epsilon > 0$ such that the number of left-side nodes n_l of $B^{(l)}$ satisifies $n_l \geq (1+\epsilon)n_{l-1}$ for all step $0 \leq l < L$ in the expansion sequence. One can take $\epsilon = reduction_frac/(1 - reduction_frac)$ for example. Thus we can bound the length of the expansion sequence by $\lceil \log_{1+\epsilon} n \rceil \in \mathcal{O}(\log n)$.

As expansion only increases the number of nodes, we are assured that all B_l have fewer than n left-side nodes and m right-side nodes. A similar guarantee does not hold for the number of edges. Indeed, nothing prevents an intermediate bipartite graph of having more than k edges that will be removed in subsequent refinement steps. Nevertheless, since during training coarsening only reduces the number of incidences, it is reasonable to expect the model to accurately refine the edges at each step instead of accumulating them, having learned the inverse of the coarsening process. Hence, for the purpose of this analysis, we assume that $k_l \leq k$ for $0 \leq l \leq L$. A similar reasoning allows to assume that $m_l \leq m$ for $0 \leq l \leq L$.

Now we bound the complexity of generating a step in the expansion sequence. For $l = L$, this requires instantiating a pair of connected nodes and predicting their expansion numbers $\mathbf{v}_L$, which is of complexity $\mathcal{O}(1)$.

For the other steps $0 \leq l < L$, starting from the bipartite representation $B^{(l+1)}$ and the expansion vectors $\mathbf{v}_L^{(l+1)}$ and $\mathbf{v}_R^{(l+1)}$, the algorithm constructs the expansion $\tilde{B}(B^{(l+1)}, \mathbf{v}_L^{(l+1)}, \mathbf{v}_R^{(l+1)})$, which is of complexity $\mathcal{O}(n+m)$. Next, it samples $\mathbf{v}_L^{(l)}$, $\mathbf{v}_R^{(l)}$ and $\mathbf{e}^{(l)}$ and constructs the refinement $B^{(l)} = B(\tilde{B}^{(l)}, \mathbf{e}^{(l)})$. Let $v_{\max}^L$ and $v_{\max}^R$ denote the maximum size of, respectively, left-side clusters and right-side clusters, which are 2 and 3 in our case. The number of incidences in the expansion $\tilde{B}^{(l)}$ can then be upper bounded by $k_l \leq k_{l+1} v_{\max}^L v_{\max}^R$.

Sampling $\mathbf{v}_L^{(l)}$, $\mathbf{v}_R^{(l)}$ and $\mathbf{e}^{(l)}$ requires querying a denoising model a fixed number of times. The underlying complexity of this process then depends on the chosen model architecture. In our setting, as a bipartite graph does not contain any triangle, the *Local PPGN* (Bergmeister et al. 2024) achieves a linear complexity in the number of nodes and incidences, *i.e.*, $\mathcal{O}(n+m+k)$.

Then, the complexity of obtaining the node embeddings for $B^{(l)}$ can be bounded by $\tilde{\mathcal{O}}(n+m+k)$. Indeed, this requires computing the K main eigenvalues and eigenvectors of the graph laplacian of $B^{(l+1)}$. Using the method of Vishnoi (2013), this is of complexity $\tilde{\mathcal{O}}\left(K k_{l+1}\right)$, where $\tilde{\mathcal{O}}$ hides polylogarithmic factors. Computing the embeddings using *SignNet* is done with complexity $\mathcal{O}\left(K(n_{l+1} + m_{l+1} + k_{l+1})\right)$, and their replication during the expansion is of complexity linear in the number of nodes and hyperedges. As K is fixed, the final complexity for computing the node embeddings is $\tilde{\mathcal{O}}(n+m+k)$.

Finally, after the expansion sequence is generated, the final bipartite representation is converted into a hypergraph by collapsing all its right-side nodes into hyperedges, which is of complexity $\mathcal{O}(m+k)$.

Combining all of the above, under the stated assumptions, the complexity to generate a hypergraph H of n nodes, m hyperedges and k incidences is of $\tilde{\mathcal{O}}((n+m+k)\log n)$, again hiding polylogarithmic factors.

C Implementation Details

C.1 Model Architecture

In our approach, we treat the expansion numbers for the left and right-side nodes and the existence of edges as attributes of the bipartite graph. We employ the EDM denoising diffusion framework (Karras et al. 2022) for modeling $p(\mathbf{v}_L^{(l)}, \mathbf{v}_R^{(l)}, \mathbf{e}^{(l)} | \tilde{B}^{(l)})$. In this framework, the target attributes—the expansion numbers and the existence of edges—are noised, and a denoising model is trained to recover their original values.

The architecture of the denoising model is as follows:

1. **Positional Encoding:** We use SignNet (Lim et al. 2022) to encode the position of each node in the graph and duplicate the encodings based on the expansion numbers.
2. **Attribute Embedding:** Three linear layers embed the attributes of the bipartite representation: one for the left-side nodes, one for the right-side nodes, one for the edges of the bipartite representation.
3. **Feature Concatenation:**
   - For both left and right-side nodes: We concatenate feature embeddings, positional encodings, desired reduction fraction, and targeted hypergraph size (*i.e.*, the desired number of left-side nodes).
   - For edges: We concatenate feature embeddings, concatenated positional encodings of the two nodes forming the edge, desired reduction fraction, and targeted hypergraph size.
4. **Graph Processing:** These three sets of vectors are then used as attributes of the bipartite graph, which is fed into a succession of sparse PPGN layers (see (Bergmeister et al. 2024)).
5. **Output Generation:** The attributes of the resulting graph after the final layer are fed to three linear layers to produce the final predictions: one for the left-side nodes, one for the right-side nodes, one for the edges.

C.2 Additional Tricks

Deterministic Expansion Size. Our expansion method typically samples two cluster size vectors $\mathbf{v}_L$ and $\mathbf{v}_R$ to enlarge the graph incrementally until $\mathbf{v}_L$ consists entirely of ones. However, this stochastic approach may not consistently yield graphs of a specified size. Following (Bergmeister et al. 2024), to address this we use a deterministic expansion strategy by predetermining the target size of the expanded graph. Instead of sampling $\mathbf{v}_L$, we select nodes with the highest expansion probabilities to achieve the desired size. We also use the proposed reduction fraction of (Bergmeister et al. 2024) as an additional input during training and inference, calculated as one minus the ratio of node counts between the original and expanded graphs. Note that this only affects $\mathbf{v}_L$, $\mathbf{v}_R$ is sampled without any limitation, playing a similar role as regular edges in the graph case. Appendix F provides further details on this approach.

Perturbed Expansion. Again following Bergmeister et al. (2024), while Definitions 8 and 9 are enough to reverse coarsening steps, we introduce additional randomness in the expansion process to improve generative performance, especially for limited datasets prone to overfitting. Our perturbed expansion randomly adds with a given probability edges between nodes of both sides of the bipartite representation that are within a predefined radius, supplementing the original edges.

The following definition formalizes the concept of a perturbed hypergraph expansion, which is a generalization of the deterministic hypergraph expansion introduced in Definition 8. A visual representation of this concept is provided in Figure 5.

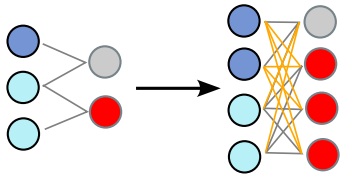


Figure 5: Depiction of a perturbed expansion. The bipartite representation $B^{(l)}$ is expanded into $\tilde{B}^{(l-1)}$ using the cluster size vectors. Deterministic expansion components are represented by gray edges, whereas additional orange edges are added for a radius of $r = 2$ and a probability of $p = 1$. With $p < 1$, a subset of these edges would be randomly excluded.

Definition 12 (Perturbed Hypergraph Expansion)**.** Given a bipartite representation $B = (\mathcal{V}_L, \mathcal{V}_R, \mathcal{E})$, two cluster size vectors $\mathbf{v}_L \in \mathbb{N}^{|\mathcal{V}_L|}$ and $\mathbf{v}_R \in \mathbb{N}^{|\mathcal{V}_R|}$, a radius $r \in \mathbb{N}$, and a probability $0 \leq p \leq 1$, the perturbed expansion $\tilde{B}$ is constructed as in Definition 8, and additionally for all distinct nodes $\mathbf{v}_L(p) \in \tilde{\mathcal{V}}_L, \mathbf{v}_R(q) \in \tilde{\mathcal{V}}_R$ whose distance in B is at most $2r + 1$, we add each edge $e_{\{p_i, q_j\}}$ independently to $\tilde{\mathcal{E}}$ with probability p.

Spectral Conditioning. Following Martinkus et al. (2022) and Bergmeister et al. (2024), we use principal *normalized* Laplacian eigenvalues and eigenvectors of the target graph as conditional information during generation, which is known to improve generation quality. When generating $B^{(l)}$ from its coarser version $B^{(l+1)}$, we exploit the spectral preservation during coarsening to approximate $B^{(l)}$'s normalized Laplacian spectrum: we compute the k smallest non-zero eigenvalues and corresponding eigenvectors of $B^{(l+1)}$'s *normalized* Laplacian matrix $\mathcal{L}^{(l+1)}$. Using SignNet (see (Lim et al. 2022)), we derive node embeddings for $B^{(l+1)}$, which are then replicated across expansion sets to initialize the embeddings of $B^{(l)}$, facilitating cluster identification. The value of k is chosen as a hyperparameter.

D Experimental Details

D.1 Datasets

Our experimental evaluation employs four synthetic and three real-world datasets:

- **Erdos-Renyi hypergraphs** (Erdős and Rényi 1960)**:** These consist of 32 nodes, with 2-edges selected with 0.1 probability, 3-edges with 0.005 probability, and 4-edges with 0.0005 probability.
- **Stochastic Block Model hypergraphs** (Kim, Bandeira, and Goemans 2018)**:** Composed of 32 nodes evenly distributed between two groups. All hyperedges are 3-edges, with inter-group edges selected with 0.001 probability and intra-group edges with 0.05 probability.
- **Ego hypergraphs** (Comrie and Kleinberg 2021)**:** Generated by first creating a hypergraph of 150-200 nodes with 3000 random hyperedges (up to 5 nodes each). A random node is then selected, and the ego hypergraph is constructed by retaining only the hyperedges containing this node.
- **Tree hypergraphs** (Nieminen and Peltola 1999)**:** We construct a tree-structured hypergraph by first generating a tree with 32 nodes using *networkx*. Subsequently, we iteratively combine adjacent edges to form hyperedges, with each hyperedge encompassing up to 5 nodes.
- **ModelNet40 meshes** (Wu et al. 2015)**:** We convert mesh topologies from selected classes of the ModelNet40 dataset into hypergraphs. To manage computational complexity, we create low-poly versions with fewer than 1000 vertices. The classes used are *bookshelf*, *piano*, and *plant*. Low-poly versions are created by iteratively merging vertices closer than a threshold until the desired vertex count is achieved. Duplicate triangles are then consolidated.

All datasets are randomly partitioned into 128 hypergraphs for training, 32 for validation, and 40 for testing.

D.2 Evaluation Metrics

We assess the generated hypergraphs against the test dataset using the following metrics:

- **NodeNumDiff:** Average difference in node count between target and generated hypergraphs.
- **NodeDegreeDistrWasserstein:** Wasserstein distance between node degree distributions of test and generated hypergraphs.
- **EdgeSizeDistrWasserstein:** Wasserstein distance between edge size distributions of test and generated hypergraphs.
- **Spectral:** Maximum Mean Discrepancy between Laplacian spectra of test and generated hypergraphs.
- **Uniqueness:** Proportion of non-isomorphic generated hypergraphs.
- **Novelty:** Proportion of generated hypergraphs that are non-isomorphic to a sample from the training set.
- **CentralityCloseness, CentralityBetweenness, CentralityHarmonic:** Wasserstein distances between distributions of each centrality measure for test and generated hypergraphs, computed on edges for $s = 1$. For details, see (Aksoy et al. 2020).

- **ValidEgo:** Applicable only to the *hypergraphEgo* dataset, measuring the proportion of generated hypergraphs that are valid ego hypergraphs (*i.e.*, containing an ego node present in all hyperedges).
- **ValidSBM:** Applicable only to the *hypergraphSBM* dataset, measuring the proportion of generated hypergraphs that are valid SBM hypergraphs (*i.e.*, containing two clusters with approximately the same probability of intra and inter-cluster hyperedges as the training set).
- **ValidTree:** Applicable only to the *hypergraphTree* dataset, measuring the proportion of generated hypergraphs that are valid tree hypergraphs.

D.3 Baseline Methods

We evaluate our method against the following baseline approaches:

- **HyperPA** (Do et al. 2020): An algorithmic method for generating hypergraphs.
- **Incidence Matrix Image-based Models**: We implement three simple baselines — Diffusion, GAN, and VAE — that generate hypergraphs via incidence matrix images:
  - These models are trained to produce images representing incidence matrices, where white pixels denote the presence of a node in a hyperedge, and black pixels indicate absence.
  - We randomly permute rows and columns, then pad with black pixels to maintain consistent dimensions.
  - A thresholding operation is applied to the generated images to obtain the final incidence matrices.

E Coarsening Sequence Sampling

This section outlines our methodology for sampling a coarsening sequence $\pi \in \Pi_F(H)$ for a given hypergraph H. Algorithm 1 presents our approach in detail.

Consider a coarsening step l where $H^{(l)}$ denotes the coarsened hypergraph, $B^{(l)}$ its bipartite representation, and $C^{(l)}$ its weighted clique expansion. The process begins by sampling a reduction fraction red_frac from the interval $[\rho_{\min}, \rho_{\max}]$. We then compute the cost of all contraction sets $F(C^{(l)})$ for the weighted clique expansion, with lower costs indicating higher contraction preference.

We use a greedy and randomized strategy, inspecting contraction sets from lowest to highest cost. For each set:

- The set is rejected with probability λ.
- If accepted:
  - **First accepted contraction:** We compute the coarsened weighted clique expansion (Definition 1) and the coarsened bipartite representation (Definition 3). The contraction is added to the set of applied contractions.
  - **Subsequent accepted contractions:** If any node in the contraction set already belongs to an accepted contraction, we reject it. Otherwise, we compute the coarsened representations based on previously accepted contractions. If all right-side clusters in the bipartite representation comprise at most three nodes of $B^{(l)}$, we add the contraction to the set of applied contractions. If not, we reject it and revert to the previous coarsened representation.

The process terminates when $|\mathcal{V}_L| - |\bar{\mathcal{V}_L}| > red_frac * |\mathcal{V}_L|$, *i.e.*, when the number of nodes in the hypergraph has been reduced by red_frac. Note that only the left-side (corresponding to the hypergraph nodes) is considered in this stopping criterion.

This approach is flexible, allowing for various choices of cost function c, contraction family F, reduction fraction range $[\rho_{\min}, \rho_{\max}]$, and randomization parameter λ.

Practical Considerations. To address the potential imbalance caused by an abundance of small graphs in the coarsening sequence, we adopt a strategy similar to (Bergmeister et al. 2024). Specifically, when the current graph has fewer than 16 nodes, we automatically set the reduction fraction ρ to its maximum value, $\rho_{\max}$. Due to the constraint that for any coarsening step, no right-side cluster can contain more than three nodes, it is not always feasible to achieve a partitioning that achieves the desired reduction fraction. Empirically, provided that $\rho_{\max}$ is small enough, this is rarely the case in practice.

During the training phase, our approach involves sampling a coarsening sequence from each dataset graph. However, we only use a graph from a randomly selected level within this sequence. As a result, the practical implementation of Algorithm 1 is designed to return a coarsened graph with its associated node and edge features, rather than the entire coarsening sequence π. To enhance computational efficiency, we kept the caching mechanism of (Bergmeister et al. 2024). Upon generating a coarsening sequence, we store its components in a cache. During training, we then randomly select a level, return the corresponding graph and features, and remove this element from the cache. This approach allows us to avoid unnecessary recomputation, as we only need to regenerate the coarsening sequence for a specific graph when all cached elements have been exhausted.

Hyperparameters. In all experiments described in Section 4, we use the following settings:

- Contraction family: The set of all edges in the clique representation, *i.e.*, $F(C) = \mathcal{E}$, for a weighted clique expansion $C = (\mathcal{V}, \mathcal{E})$.
- Cost function: Local Variation Cost (Loukas 2019) with a preserving eigenspace size of $k = 8$.
- Reduction fraction range: $[\rho_{\min}, \rho_{\max}] = [0.1, 0.3]$.
- Randomization parameter: $\lambda = 0.3$.

Algorithm 1 Hypergraph coarsening sequence sampling: This describes the randomized iterative coarsening of a hypergraph. At each step, contraction sets are selected based on their cost and a sampled reduction fraction, while ensuring that right-side clusters contain at most three nodes. The process continues until the bipartite representation is reduced to a single pair of connected nodes.

Parameters: contraction family F, cost function c, reduction fraction range $[\rho_{\min}, \rho_{\max}]$, randomization parameter λ
Input: hypergraph H
Output: coarsening sequence $\pi = (B^{(0)}, \ldots, B^{(L)})$

function HYPERGRAPHCOARSENINGSEQ(H)
  $H^{(0)} \leftarrow H$, $B^{(0)} \leftarrow$ BipartiteRepresentation($H^{(0)}$), $C^{(0)} \leftarrow$ WeightedCliqueExpansion($H^{(0)}$)
  $\pi \leftarrow (B^{(0)})$
  $l \leftarrow 0$
  while $|\mathcal{V}_L^{(l)}| > 1$ **do**
    $l \leftarrow l + 1$
    red_frac $\sim$ Uniform($[\rho_{\min}, \rho_{\max}]$) ▷ random reduction fraction
    $f \leftarrow c(\cdot, C^{(l-1)})$ ▷ cost function for clique expansion
    accepted_contractions $\leftarrow \emptyset$
    $C_{\text{next}} \leftarrow C^{(l-1)}$
    $B_{\text{next}} \leftarrow B^{(l-1)}$
    for $S \in$ SortedByCost($F(C^{(l-1)}), f$) **do**
      if Random() $> \lambda$ **then**
        if $S \cap (\bigcup_{P \in \text{accepted_contractions}} P) = \emptyset$ **then**
          $C_{\text{temp}} \leftarrow$ CoarsenCliqueExpansion(C_{next}, S)
          $B_{\text{temp}} \leftarrow$ CoarsenBipartite(B_{next}, S)
          if $\forall$ right cluster $R \in B_{\text{temp}} : |R| \leq 3$ **then**
            accepted_contractions $\leftarrow$ accepted_contractions $\cup \{S\}$
            $C_{\text{next}} \leftarrow C_{\text{temp}}$
            $B_{\text{next}} \leftarrow B_{\text{temp}}$
          end if
        end if
      end if
      if $|\mathcal{V}_L^{(l-1)}| - |\mathcal{V}_{L,\text{next}}| >$ red_frac $\cdot |\mathcal{V}_L^{(l-1)}|$ **then**
        break
      end if
    end for
    $C^{(l)} \leftarrow C_{\text{next}}$, $B^{(l)} \leftarrow B_{\text{next}}$
    $\pi \leftarrow \pi \cup \{B^{(l)}\}$
  end while
  return π
end function

F End-to-end Training and Sampling

In this section, we detail the end-to-end training and sampling procedures in Algorithms 3 and 4, respectively. Both assume the deterministic expansion size setting introduced in Section C.2. When deterministic expansion size is disabled, the expansion sizes are sampled directly from the model predictions rather than being selected to match a prescribed target expansion size; the corresponding sampling procedure is given in Algorithm 5. All algorithms rely on the node embedding computation defined in Algorithm 2.

Algorithm 2 Node embedding computation: This describes how node embeddings are computed for a given bipartite representation of a hypergraph.

Parameters: number of spectral features k
Input: bipartite representation $B = (\mathcal{V}_L, \mathcal{V}_R, \mathcal{E})$, spectral feature model SignNet_θ, cluster size vectors $\mathbf{v}_L$ and $\mathbf{v}_R$
Output: node embeddings computed for all nodes in $\mathcal{V}_L$ and $\mathcal{V}_R$ and replicated according to $\mathbf{v}_L$ and $\mathbf{v}_R$

function EMBEDDINGS($B = (\mathcal{V}_L, \mathcal{V}_R, \mathcal{E})$, SignNet_θ, $\mathbf{v}_L$, $\mathbf{v}_R$)
  $n_B \leftarrow |\mathcal{V}_L| + |\mathcal{V}_R|$
  if $k = 0$ **then**
    $\mathbf{H} = [h^{(1)}, \ldots, h^{(n_B)}], \quad h^{(i)} \overset{i.i.d.}{\sim} \mathcal{N}(0, I)$ ▷ Sample random embeddings
  else
    $k_{\text{eff}} \leftarrow \min(k, n_B - 2)$
    if $k_{\text{eff}} > 0$ **then**
      $[\lambda_1, \ldots, \lambda_{k_{\text{eff}}}], [u_1, \ldots, u_{k_{\text{eff}}}] \leftarrow \text{EIG}(B)$ ▷ Compute smallest non-trivial spectral features
    end if
    if $k_{\text{eff}} < k$ **then**
      $[\lambda_{k_{\text{eff}}+1}, \ldots, \lambda_k], [u_{k_{\text{eff}}+1}, \ldots, u_k] \leftarrow [0, \ldots, 0], [0, \ldots, 0]$ ▷ Pad with zeros
    end if
    $\mathbf{H} = [h^{(1)}, \ldots, h^{(n_B)}] \leftarrow \text{SignNet}_\theta([\lambda_1, \ldots, \lambda_k], [u_1, \ldots, u_k], B)$
  end if
  $\tilde{B} = (\mathcal{V}_L^{(1)} \cup \cdots \cup \mathcal{V}_L^{(|\mathcal{V}_L|)}, \mathcal{V}_R^{(1)} \cup \cdots \cup \mathcal{V}_R^{(|\mathcal{V}_R|)}, \tilde{\mathcal{E}}) \leftarrow \tilde{B}(B, \mathbf{v}_L, \mathbf{v}_R)$ ▷ Expand as per Definition 8
  Replicate embeddings according to cluster sizes:

$$\tilde{\mathbf{H}}[v_L^{(p,i)}] = \mathbf{H}[v_L^{(p)}], \quad \forall p \in [|\mathcal{V}_L|],\ 1 \le i \le \mathbf{v}_L[p]$$

$$\tilde{\mathbf{H}}[v_R^{(p,i)}] = \mathbf{H}[v_R^{(p)}], \quad \forall p \in [|\mathcal{V}_R|],\ 1 \le i \le \mathbf{v}_R[p]$$

  return $\tilde{\mathbf{H}}$
end function

Algorithm 3 End-to-end training procedure: This describes the entire training procedure for our model.

Parameters: number of spectral features k for node embeddings
Input: dataset $\mathcal{D} = \{H_1, \ldots, H_N\}$, denoising model GNN_θ, spectral feature model SignNet_θ
Output: trained model parameters θ

function TRAIN($\mathcal{D}$, GNN_θ, SignNet_θ)
  while not converged **do**
    $H \sim \text{Uniform}(\mathcal{D})$ ▷ Sample graph
    $(B^{(0)}, \ldots, B^{(L)}) \leftarrow \text{RndRedSeq}(H)$ ▷ Sample coarsening sequence by Algorithm 1
    $l \sim \text{Uniform}(\{0, \ldots, L\})$ ▷ Sample level
    if $l = 0$ **then**
      $\mathbf{v}_L^{(0)} \leftarrow 1$, $\mathbf{v}_R^{(0)} \leftarrow 1$
    else
      Set $\mathbf{v}_L^{(l)}$ and $\mathbf{v}_R^{(l)}$ as in (9), s.t. the node sets of $\tilde{B}(B^{(l)}, \mathbf{v}_L^{(l)}, \mathbf{v}_R^{(l)})$ equal those of $B^{(l-1)}$
    end if
    if $l = L$ **then**
      $B^{(L+1)} \leftarrow (\{1\}, \{2\}, \{(1,2)\})$
      $\mathbf{v}_L^{(L+1)} \leftarrow 1$
      $\mathbf{v}_R^{(L+1)} \leftarrow 1$
    else
      Set $\mathbf{v}_L^{(l+1)}$ and $\mathbf{v}_R^{(l+1)}$ as in (9), s.t. the node sets of $\tilde{B}(B^{(l+1)}, \mathbf{v}_L^{(l+1)}, \mathbf{v}_R^{(l+1)})$ equal those of $B^{(l)}$
    end if
    Set $\mathbf{e}^{(l)}$ as in Eq. (10), s.t. $B\left(\tilde{B}(B^{(l+1)}, \mathbf{v}_L^{(l+1)}, \mathbf{v}_R^{(l+1)}), \mathbf{e}^{(l)}\right) = B^{(l)}$
    $\mathbf{H}^{(l)} \leftarrow \text{Embeddings}(B^{(l+1)}, \text{SignNet}_\theta, \mathbf{v}_L^{(l+1)}, \mathbf{v}_R^{(l+1)})$ ▷ Compute node embeddings
    $\hat{\rho}^{(l)} \leftarrow 1 - \dfrac{|\mathcal{V}_L^{(l+1)}|}{|\mathcal{V}_L^{(l)}|}$
    $D_\theta \leftarrow \text{GNN}_\theta(\cdot, \cdot, \tilde{B}^{(l)}, \mathbf{H}^{(l)}, |\mathcal{V}_L^{(0)}|, \hat{\rho}^{(l)})$
    Take gradient descent step on $\nabla_\theta \text{DiffusionLoss}(\mathbf{v}_L^{(l)}, \mathbf{v}_R^{(l)}, \mathbf{e}^{(l)}, D_\theta)$
  end while
  return θ
end function

Algorithm 4 End-to-end sampling procedure with deterministic expansion: This describes the sampling procedure with deterministic expansion, described in Section C.2. Note that this assumes that maximum cluster sizes are 2 for nodes and 3 for hyperedges, which is the case when using edges of the clique representation as the contraction set family.

Parameters: reduction fraction range $[\rho_{\min}, \rho_{\max}]$
Input: target hypergraph size N, denoising model GNN_θ, spectral feature model SignNet_θ
Output: sampled hypergraph $H = (\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}| = N$

function SAMPLE(N, GNN_θ, SignNet_θ)
  $B = (\mathcal{V}_L, \mathcal{V}_R, \mathcal{E}) \leftarrow (\{1\}, \{2\}, \{(1, 2)\})$ ▷ Start with a minimal bipartite graph
  $\mathbf{v}_L \leftarrow [1]$, $\mathbf{v}_R \leftarrow [1]$ ▷ Initial cluster size vectors
  while $|\mathcal{V}_L| < N$ **do**
    $\mathbf{H} \leftarrow \text{Embeddings}(B, \text{SignNet}_\theta, \mathbf{v}_L, \mathbf{v}_R)$ ▷ Compute node embeddings
    $\tilde{B} \leftarrow \tilde{B}(B, \mathbf{v}_L, \mathbf{v}_R)$
    $n \leftarrow \|\mathbf{v}_L\|_1$
    $\rho \sim \text{Uniform}([\rho_{\min}, \rho_{\max}])$ ▷ random reduction fraction
    $n_{\text{next}} \leftarrow \min\left(\max\left(\left\lceil \frac{n}{1-\rho} \right\rceil, n+1\right), N\right), \qquad n^+ \leftarrow n_{\text{next}} - n$ ▷ number of left-side nodes to add
    $\hat{\rho} \leftarrow 1 - \dfrac{n}{n_{\text{next}}}$ ▷ actual reduction fraction
    $D_\theta \leftarrow \text{GNN}_\theta(\cdot, \cdot, \tilde{B}, \mathbf{H}, N, \hat{\rho})$
    $(\mathbf{v}_L)_0, (\mathbf{v}_R)_0, (\mathbf{e})_0 \leftarrow \text{Sample}(D_\theta)$ ▷ Sample features
    Set $\mathbf{v}_L$ s.t. for $i \in [n]$:

$$\mathbf{v}_L[i] = \begin{cases} 2 & \text{if } |\{j \in [n] \mid (\mathbf{v}_L)_0[j] \geq (\mathbf{v}_L)_0[i]\}| \leq n^+, \\ 1 & \text{otherwise} \end{cases}$$

    Set $\mathbf{v}_R$ s.t. for $i \in [|(\mathbf{v}_R)_0|]$:

$$\mathbf{v}_R[i] = \begin{cases} 1 & \text{if } (\mathbf{v}_R)_0[i] < 1.66, \\ 2 & \text{if } 1.66 \leq (\mathbf{v}_R)_0[i] < 2.33, \\ 3 & \text{otherwise} \end{cases}$$

    Set $\mathbf{e}$ s.t. for $i \in [|(\mathbf{e})_0|]$:

$$\mathbf{e}[i] = \begin{cases} 1 & \text{if } (\mathbf{e})_0[i] > 0.5, \\ 0 & \text{otherwise} \end{cases}$$

    $B = (\mathcal{V}_L, \mathcal{V}_R, \mathcal{E}) \leftarrow B(\tilde{B}, \mathbf{e})$ ▷ Refine as per Definition 9
  end while
  Build H from its bipartite representation B
  return H
end function

Algorithm 5 End-to-end sampling procedure without deterministic expansion: This describes the entire sampling procedure without deterministic expansion. Note that this assumes that maximum cluster sizes are 2 for nodes and 3 for hyperedges, which is the case when using edges of the clique representation as the contraction set family.

Parameters: reduction fraction range $[\rho_{\min}, \rho_{\max}]$
Input: target hypergraph size N, denoising model GNN_θ, spectral feature model SignNet_θ
Output: sampled hypergraph $H = (\mathcal{V}, \mathcal{E})$

function SAMPLE(N, GNN_θ, SignNet_θ)
  $B = (\mathcal{V}_L, \mathcal{V}_R, \mathcal{E}) \leftarrow (\{1\}, \{2\}, \{(1, 2)\})$ ▷ Start with a minimal bipartite graph
  $\mathbf{v}_L \leftarrow [1], \mathbf{v}_R \leftarrow [1]$ ▷ Initial cluster size vectors
  while $|\mathcal{V}_L| < N$ **do**
    $\mathbf{H} \leftarrow \text{Embeddings}(B, \text{SignNet}_\theta, \mathbf{v}_L, \mathbf{v}_R)$ ▷ Compute node embeddings
    $\tilde{B} \leftarrow \tilde{B}(B, \mathbf{v}_L, \mathbf{v}_R)$
    $n \leftarrow \|\mathbf{v}_L\|_1$
    $\rho \sim \text{Uniform}([\rho_{\min}, \rho_{\max}])$ ▷ random reduction fraction
    $n_{\text{next}} \leftarrow \min\left(\max\left(\left\lceil \frac{n}{1-\rho} \right\rceil, n+1\right), N\right)$
    $\hat{\rho} \leftarrow 1 - \dfrac{n}{n_{\text{next}}}$ ▷ conditioning reduction fraction
    $D_\theta \leftarrow \text{GNN}_\theta(\cdot, \cdot, \tilde{B}, \mathbf{H}, N, \hat{\rho})$
    $(\mathbf{v}_L)_0, (\mathbf{v}_R)_0, (\mathbf{e})_0 \leftarrow \text{SDESample}(D_\theta)$ ▷ Sample feature embeddings
    et $\mathbf{v}_L$ s.t. for $i \in [|(\mathbf{v}_L)_0|]$:

$$\mathbf{v}_L[i] = \begin{cases} 1 & \text{if } (\mathbf{v}_L)_0[i] < 1.5, \\ 2 & \text{otherwise} \end{cases}$$

    Set $\mathbf{v}_R$ s.t. for $i \in [|(\mathbf{v}_R)_0|]$:

$$\mathbf{v}_R[i] = \begin{cases} 1 & \text{if } (\mathbf{v}_R)_0[i] < 1.66, \\ 2 & \text{if } 1.66 \leq (\mathbf{v}_R)_0[i] < 2.33, \\ 3 & \text{otherwise} \end{cases}$$

    Set $\mathbf{e}$ s.t. for $i \in [|(\mathbf{e})_0|]$:

$$\mathbf{e}[i] = \begin{cases} 1 & \text{if } (\mathbf{e})_0[i] > 0.5, \\ 0 & \text{otherwise} \end{cases}$$

    $B = (\mathcal{V}_L, \mathcal{V}_R, \mathcal{E}) \leftarrow B(\tilde{B}, \mathbf{e})$ ▷ Refine as per Definition 9
  end while
  Build H from its bipartite representation B
  return H
end function

G Detailed Results

Model	Erdos-Renyi Hypergraphs ($n_{avg} = 32$, $std = 0.07$)									SBM Hypergraphs ($n_{avg} = 31.73$, $std = 0.55$)									
	Node Num ↓	Node Deg ↓	Edge Size ↓	Spectral ↓	Uniq. ↑	Nov. ↑	Cent. Close ↓	Cent. Betw. ↓	Cent. Harm. ↓	Valid SBM ↑	Node Num ↓	Node Deg ↓	Edge Size ↓	Spectral ↓	Uniq. ↑	Nov. ↑	Cent. Close ↓	Cent. Betw. ↓	Cent. Harm. ↓
HyperPA	**0.000**	5.530	0.183	0.177	**1**	**1**	0.078	0.014	107.1	2.5%	**0.075**	4.062	0.407	0.273	**1**	**1**	0.074	0.008	77.84
VAE	0.100	2.140	0.540	0.035	**1**	**1**	0.079	0.008	13.50	0%	0.375	1.280	1.059	0.024	**1**	**1**	**0.007**	0.006	6.543
GAN	0.675	2.560	0.657	0.048	**1**	**1**	0.101	0.011	17.16	0%	1.200	2.106	1.203	0.059	**1**	**1**	0.076	0.012	10.70
Diffusion	0.050	2.225	0.781	0.014	**1**	**1**	0.048	0.003	11.53	0%	0.150	1.717	1.390	0.031	**1**	**1**	0.040	0.004	13.94
HYGENE	0.775	**0.475**	**0.012**	**0.006**	**1**	**1**	**0.009**	**2.6e-4**	**2.127**	**65%**	0.525	**0.321**	**0.002**	**0.010**	**1**	**1**	0.016	**4.4e-4**	**2.990**

Table 4: Evaluation metrics for Erdos-Renyi and SBM hypergraphs

Model	Ego Hypergraphs ($n_{avg} = 109.71$, $std = 10.23$)										Tree hypergraphs ($n_{avg} = 32$, $std = 0$)									
	Valid Ego ↑	Node Num ↓	Node Deg ↓	Edge Size ↓	Spectral ↓	Uniq. ↑	Nov. ↑	Cent. Close ↓	Cent. Betw. ↓	Cent. Harm. ↓	Valid Tree ↑	Node Num ↓	Node Deg ↓	Edge Size ↓	Spectral ↓	Uniq. ↑	Nov. ↑	Cent. Close ↓	Cent. Betw. ↓	Cent. Harm. ↓
HyperPA	0%	35.83	2.590	0.423	0.237	**1**	**1**	0.354	0.002	143.0	0%	2.350	0.315	0.284	0.159	**1**	**1**	0.477	0.168	5.941
VAE	0%	47.58	0.803	1.458	0.133	**1**	**1**	0.558	0.019	38.95	0%	9.700	0.072	0.480	0.124	**1**	**1**	0.280	0.139	3.869
GAN	0%	60.35	0.917	1.665	0.230	**1**	**1**	0.612	0.015	41.80	0%	6.0	0.151	0.469	0.089	**1**	**1**	0.201	0.124	2.198
Diffusion	0%	**4.475**	3.984	2.985	0.190	**1**	**1**	0.407	0.009	6.911	0%	2.225	1.718	1.922	0.127	**1**	**1**	0.353	0.139	8.565
Ours	**90%**	12.55	**0.063**	**0.220**	**0.004**	**1**	**1**	**0.025**	**8.95e-5**	**5.790**	**77.5%**	**0**	**0.059**	**0.108**	**0.012**	**1**	**1**	**0.041**	**0.016**	**1.099**

Table 5: Evaluation metrics for Ego and Tree hypergraphs

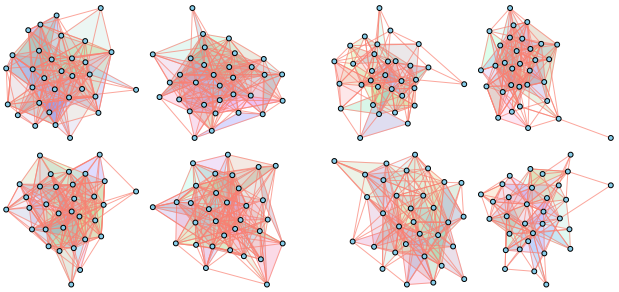
Model	ModelNet40 Plant ($n_{avg} = 124.86$, $std = 87.88$)									ModelNet40 Bookshelf ($n_{avg} = 119.38$, $std = 68.20$)								
	Node Num ↓	Node Deg ↓	Edge Size ↓	Spectral ↓	Uniq. ↑	Nov. ↑	Cent. Close ↓	Cent. Betw. ↓	Cent. Harm. ↓	Node Num ↓	Node Deg ↓	Edge Size ↓	Spectral ↓	Uniq. ↑	Nov. ↑	Cent. Close ↓	Cent. Betw. ↓	Cent. Harm. ↓
HyperPA	10.82	6.566	0.046	0.061	**1**	**1**	0.266	0.009	932.9	8.025	7.562	0.044	**0.048**	**1**	**1**	0.211	0.005	877.5
VAE	76.15	3.895	1.573	0.205	**1**	**1**	**0.230**	**0.005**	**73.50**	47.45	6.190	1.520	0.190	**1**	**1**	**0.145**	**0.003**	113.6
GAN	**0**	378.1	56.35	0.364	**1**	**1**	0.782	0.012	644.8	**0.000**	397.2	46.30	0.476	**1**	**1**	0.707	0.007	670.1
Diffusion	0.025	21.03	3.439	0.069	**1**	**1**	0.319	0.010	270.2	**0.000**	20.36	2.346	0.079	**1**	**1**	0.239	0.006	264.1
Ours	68.38	**2.428**	**0.027**	**0.034**	**1**	**1**	0.263	0.009	197.1	69.73	**1.050**	**0.034**	0.068	**1**	**1**	0.204	0.004	**27.40**

Table 6: Evaluation metrics for ModelNet40 Plant and ModelNet40 Bookshelf

Model	ModelNet40 Piano ($n_{avg} = 177.29$, $std = 57.11$)								
	Node Num ↓	Node Deg ↓	Edge Size ↓	Spectral ↓	Uniq. ↑	Nov. ↑	Cent. Close ↓	Cent. Betw. ↓	Cent. Harm. ↓
HyperPA	0.825	9.254	**0.023**	**0.067**	**1**	**1**	**0.236**	0.004	**77.84**
VAE	75.35	8.060	1.686	0.396	**1**	**1**	0.241	0.003	184.3
GAN	**0.000**	409.0	86.38	0.697	**1**	**1**	0.738	0.005	622.2
Diffusion	0.050	20.90	4.192	0.113	**1**	**1**	0.303	0.004	289.3
Ours	42.52	**6.290**	0.027	0.117	**1**	**1**	0.285	**0.002**	155.0

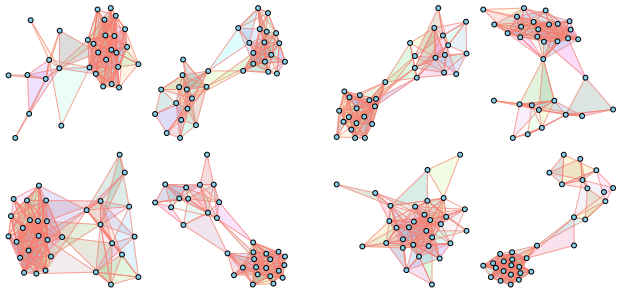
Table 7: Evaluation metrics for ModelNet40 Piano

H Comparison Between Training and Generated Samples



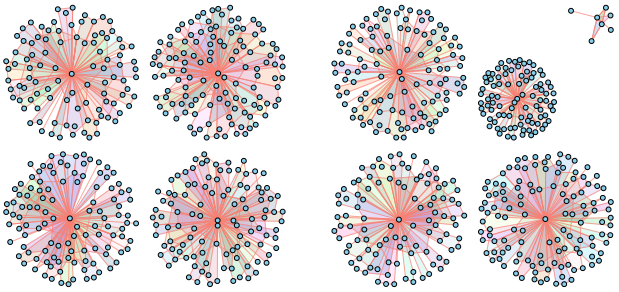
Train samples Generated samples

(a) Erdos-Renyi hypergraphs



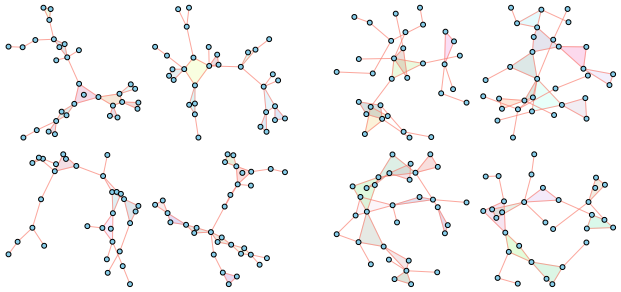
Train samples Generated samples

(b) Stochastic Block Model hypergraphs



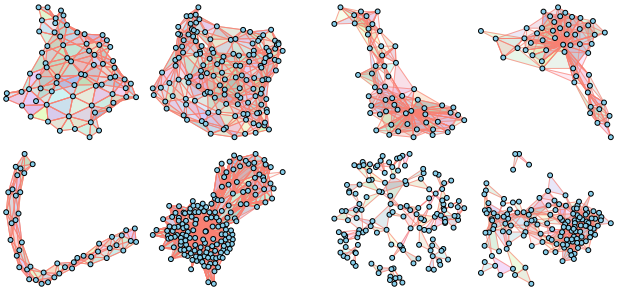
Train samples Generated samples

(c) Ego hypergraphs



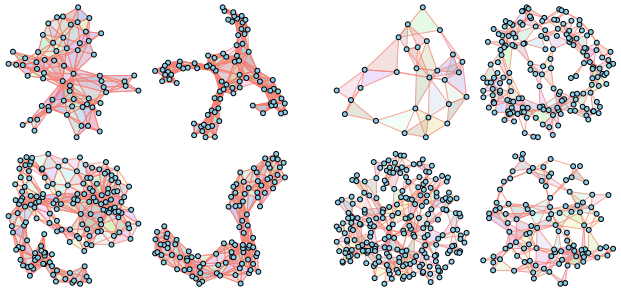
Train samples Generated samples

(d) Tree hypergraphs



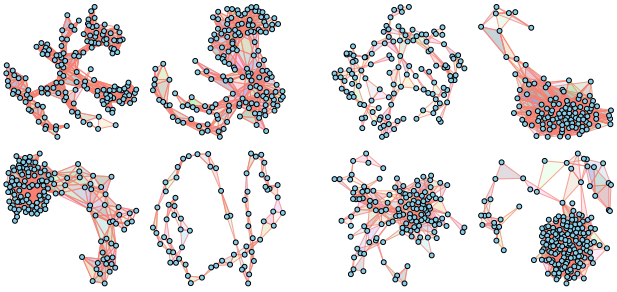
Train samples Generated samples

(e) Plant meshes topology



Train samples Generated samples

(f) Bookshelf meshes topology



Train samples Generated samples

(g) Piano meshes topology

Figure 6: Comparison of train and generated samples for various datasets