

FEATURE-AWARE (HYPER)GRAPH GENERATION VIA NEXT-SCALE PREDICTION

Anonymous authors

Paper under double-blind review

ABSTRACT

Graph generative models have shown strong results in molecular design but struggle to scale to large, complex structures. While hierarchical methods improve scalability, they usually ignore node and edge features, which are critical in real-world applications. This issue is amplified in hypergraphs, where hyperedges capture higher-order relationships among multiple nodes. Despite their importance in domains such as 3D geometry, molecular systems, and circuit design, existing generative models rarely support both hypergraphs and feature generation at scale. In this paper, we introduce FAHNES (feature-aware hypergraph generation via next-scale prediction), a hierarchical framework that jointly generates hypergraph topology and features. FAHNES builds multi-scale representations through node coarsening and refines them via localized expansion, guided by a novel node budget mechanism that controls granularity and ensures consistency across scales. Experiments on synthetic, 3D mesh and graph point cloud datasets show that FAHNES achieves state-of-the-art performance in jointly generating features and structure, advancing scalable hypergraph and graph generation.

1 INTRODUCTION

Generating discrete geometric structures such as graphs and hypergraphs is an important challenge in modern machine learning (Zhu et al., 2022). These structures are central to applications ranging from molecular design and materials discovery to electronic circuits and 3D shape modeling (Kajino, 2019; Rahman et al., 2012; Starostin & Balashov, 2008; Luo et al., 2024). Their ability to capture complex relationships—pairwise in graphs, and multi-way in hypergraphs—makes them an essential tool for representing and synthesizing structured data. In many real-world settings, topology alone is not enough: node and edge (or hyperedge) features often carry important semantic or geometric information, such as atom and bond types in molecules, or vertex coordinates in 3D meshes.

Despite recent advances, existing methods for featured graph generation struggle to scale. Most of these approaches use flat architectures that model the entire structure at once, leading to quadratic computational and memory complexities (Vignac et al., 2023; Eijkelboom et al., 2024; Xu et al., 2024). This limits their use to moderately sized graphs. To address scalability, hierarchical generation strategies have been proposed for both graphs and hypergraphs (Bergmeister et al., 2024; Gailhard et al., 2025). These methods build the structure in stages, starting from a coarse representation and progressively upsampling and refining it, which allows them to handle much larger topologies.

However, existing hierarchical methods only focus on unfeatured structures (Bergmeister et al., 2024; Gailhard et al., 2025). Extending them to generate features is challenging because different regions of a graph or hypergraph often grow at uneven rates during the refinement process, making it difficult to maintain consistency across scales. Furthermore, sequentially generating topology first and then features—rather than modeling them jointly—is ineffective in complex settings, as illustrated in Figure 1.

To overcome these limitations, we propose FAHNES (feature-aware hypergraph generation

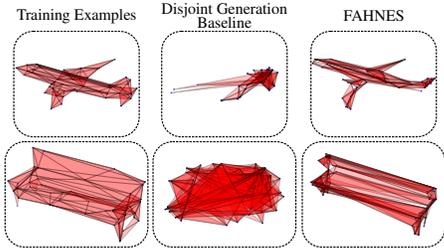


Figure 1: Examples of generated featured hypergraphs by a sequential disjoint generation baseline and our model (FAHNES).

via next-scale prediction), a hierarchical generative framework that jointly models topology and features for both graphs and hypergraphs¹. FAHNES builds multi-scale representations (Tian et al., 2024) through node coarsening and reconstructs fine structures via localized expansion, while directly predicting features alongside structure at each stage. A fundamental component of our model is a *node budget mechanism*, which encodes local growth constraints and helps maintain consistency in regions that expand at different rates. In addition, we adapt minibatch *flow-matching optimal-transport* (OT) coupling (Tong et al., 2024; Pooladian et al., 2023) to the hierarchical setting, improving stability and coherence during generation. This combination enables scalable generation of large, featured graphs or hypergraphs across diverse domains, from 3D meshes to point clouds. Our main contributions are:

- We introduce the first scalable hierarchical model for the joint generation of hypergraph topology and features, targeting complex data such as 3D meshes and point clouds.
- We propose a novel budget-based mechanism that enhances global structural coherence in hierarchical generation.
- We extend minibatch OT-coupling to hierarchical graph or hypergraph generation, improving generative stability and quality.
- We validate FAHNES on both synthetic and real-world datasets, demonstrating strong performance in jointly generating topology and features.

2 RELATED WORK

Graph and hypergraph generation using deep learning. Graph generation has seen significant advances in recent years. Early approaches, such as GraphVAE (Simonovsky & Komodakis, 2018), employed autoencoders to embed graphs into latent spaces for sampling. Subsequent models leveraged recurrent neural networks to sequentially generate adjacency matrices, improving structural fidelity (You et al., 2018). More recently, diffusion-based methods have enabled permutation-invariant graph generation (Niu et al., 2020; Vignac et al., 2023), with extensions incorporating structural priors such as node degrees (Chen et al., 2023). Many of these previous methods jointly model node features and topology, but are limited to small graphs due to scalability challenges. Diffusion-based models (Vignac et al., 2023; Eijkelboom et al., 2024; Xu et al., 2024) operate over complete graphs by gradually corrupting structures and features, then training models to denoise them. However, their scalability is constrained by the combinatorial number of possible edges in graphs.

To mitigate this, hierarchical methods have been proposed. Bergmeister et al. (2024) introduced a scalable graph generation framework based on a coarsen-then-expand approach, merging nodes to form coarse representations and progressively reconstructing finer details. This framework was extended to hypergraphs by Gailhard et al. (2025), which allows edges to connect more than two nodes. However, both methods focus exclusively on topology generation, neglecting node and hyperedge features essential for many applications.

Applications of hypergraph generation. Generative models that capture both higher-order structure and node/hyperedge features are critical in numerous domains. In molecular design, regular graph generative models struggle to accurately represent rings and scaffolds (Vignac et al., 2023; Barsbey et al., 2025), which naturally correspond to hyperedges involving multi-atom interactions rather than pairwise bonds. Similarly, 3D shape modeling involves surfaces such as triangles, quads, or general polygons that extend beyond simple pairwise connectivity. Conventional approaches often rely on fixed topology, quantization, or autoregressive sequence modeling with transformers (Nash et al., 2020; Siddiqui et al., 2024), limiting their flexibility and scalability. Hypergraphs provide a more general framework by treating faces as hyperedges, enabling the joint generation of topology and features.

Unlike prior work that focuses solely on topology or relies on flat, non-scalable designs, we present the first unified, scalable approach for jointly modeling hypergraph structure and features. FAHNES enhances hierarchical generative modeling with feature integration, a novel node-budget mechanism, and flow-matching training tailored to our strategy.

¹Since hypergraphs are a generalization of graphs, we present FAHNES on hypergraphs. The explanation for graphs follows straightforwardly.

3 FEATURE-AWARE (HYPER)GRAPH GENERATION VIA NEXT-SCALE PREDICTION

Notations. Throughout this paper, we use calligraphic letters, like \mathcal{V} , to represent sets, with their cardinality denoted by $|\mathcal{V}|$. Matrices are represented by bold uppercase letters (e.g., \mathbf{A}), while vectors are indicated by bold lowercase letters (e.g., \mathbf{x}). The transpose and point-wise multiplication operations are denoted by $(\cdot)^\top$ and \odot , respectively. $\lceil \cdot \rceil$ denotes rounding to the nearest integer.

Basic definitions. We define a graph $G = (\mathcal{V}, \mathcal{E})$ as a pair consisting of a set of vertices \mathcal{V} and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Graphs may also carry node and edge features, represented by matrices $\mathbf{F}_{\mathcal{V}} \in \mathbb{R}^{|\mathcal{V}| \times m}$ and $\mathbf{F}_{\mathcal{E}} \in \mathbb{R}^{|\mathcal{E}| \times l}$, where m and l denote the dimensionality of the features. Each edge $e \in \mathcal{E}$ corresponds to a pair (u, v) , indicating a connection between nodes u and v . A bipartite graph $B = (\mathcal{V}_L, \mathcal{V}_R, \mathcal{E})$ is a special case of a graph where the vertex set is split into two disjoint subsets \mathcal{V}_L and \mathcal{V}_R , and edges exist only between the two parts, i.e., $\mathcal{E} \subseteq \mathcal{V}_L \times \mathcal{V}_R$. The full set of nodes is $\mathcal{V} = \mathcal{V}_L \cup \mathcal{V}_R$. In this work, we consider node features for bipartite graphs, denoted by \mathbf{F}_L for the left-side nodes and \mathbf{F}_R for the right-side nodes.

A hypergraph H extends the concept of a graph and is specified by a pair $(\mathcal{V}, \mathcal{E})$, where \mathcal{V} represents the vertex set and \mathcal{E} comprises hyperedges, with each $e \in \mathcal{E}$ being a subset of \mathcal{V} . The main distinction of hypergraphs lies in their ability to connect arbitrary numbers of vertices through a single hyperedge. Similar to graphs, hypergraphs can possess node and hyperedge features, which are denoted $\mathbf{F}_{\mathcal{V}}$ and $\mathbf{F}_{\mathcal{E}}$. We consider two fundamental graph-based representations of hypergraphs: clique and star expansions. The clique expansion transforms a hypergraph H into a graph $C = (\mathcal{V}, \mathcal{E}_c)$, where hyperedges are replaced by cliques, i.e., $\mathcal{E}_c = \{(u, v) \mid \exists e \in \mathcal{E} : u, v \in e\}$. The star expansion converts a hypergraph H into a bipartite structure $B = (\mathcal{V}_L, \mathcal{V}_R, \mathcal{E}_b)$, where $\mathcal{V}_L = \mathcal{V}$, $\mathcal{V}_R = \mathcal{E}$, and $\mathcal{E}_b = \{(v, e) \mid v \in \mathcal{V}_L, e \in \mathcal{V}_R, v \in e \text{ in } H\}$. Left side nodes \mathcal{V}_L represent the nodes of the hypergraph, while right side nodes \mathcal{V}_R represent hyperedges.

Our objective is to develop a generative model that can sample from the underlying distribution of a dataset of featured hypergraphs (or graphs) (H_1, \dots, H_N) , i.e., to learn the joint distribution over both topology and features. We present FAHNES in the more general setting of hypergraphs; since graphs are a special case of hypergraphs, the model can be directly adapted to graphs, as discussed in Section 3.6. Complete mathematical proofs of all propositions in this paper appear in Appendix A.

3.1 OVERVIEW

FAHNES follows a hierarchical pipeline as shown in Figure 2: *i*) during training, multi-scale representations of input (hyper)graphs are produced; *ii*) a model learns to reconstruct a higher scale from a lower one. Our approach adopts a coarsening–expansion framework (Gailhard et al., 2025). We first perform standard spectrum-preserving coarsening (Loukas, 2019) on the clique expansion of the hypergraph to obtain the multi-scale representation. The process is then reversed through expansion and refinement, reconstructing both topology and features at progressively finer scales. Expansion and refinement are learned on the bipartite representation using the flow-matching framework (Lipman et al., 2022), treating generation as the inverse of the coarsening process.

To address uneven growth across regions, we introduce a *node budget* mechanism: each cluster is assigned a budget indicating its remaining expansions. Budgets are recursively divided among child nodes, starting from a single super-node with the full budget and ending when all clusters have a budget of one. This provides more local control over the final node count and improves global consistency compared to prior methods (Bergmeister et al., 2024; Gailhard et al., 2025), which only append the desired size to node embeddings. At each expansion step, the model also predicts the mean feature of future child nodes conditioned on the parent’s feature, extending the scale-wise autoregressive idea of Ren et al. (2024) so that predictions at one scale guide those at the next.

3.2 BUDGETED COARSENING

We adopt the coarsening strategy of Gailhard et al. (2025), representing the hypergraph at multiple resolution levels by merging nodes into clusters. Node pairs to be merged are selected using the spectrum-preserving coarsening of Loukas (2019), applied to the hypergraph’s clique expansion. Mergings in the clique expansion are then mirrored in the bipartite representation, with corresponding

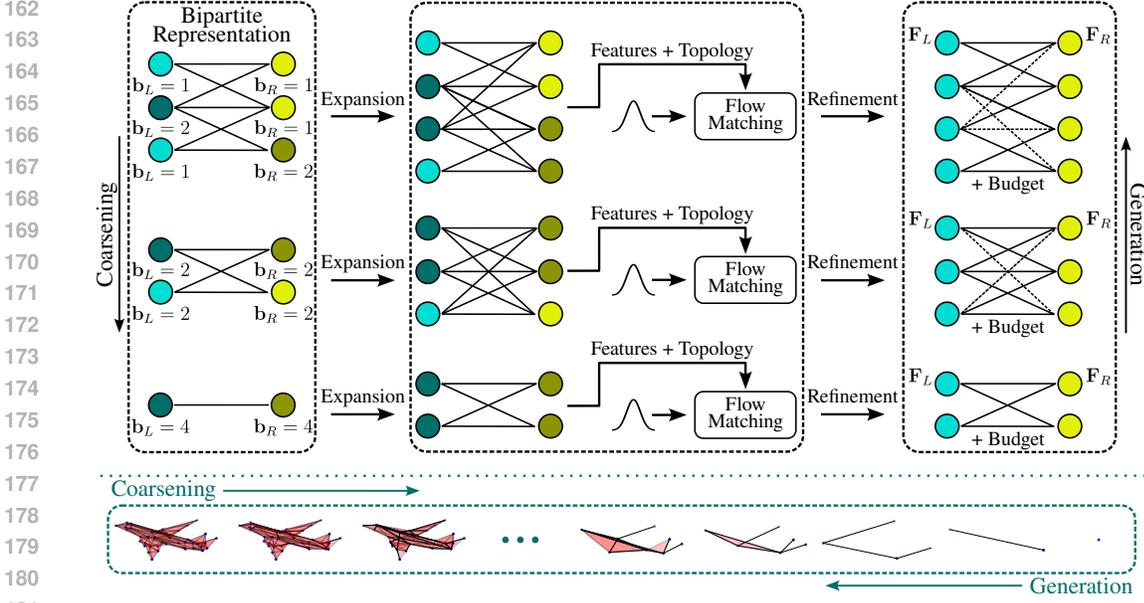


Figure 2: Our framework adopts a *coarsening-expansion* strategy. *i*) During training, input hypergraphs are progressively coarsened by merging nodes and hyperedges, yielding a multiscale representation. Node features are averaged during merging, and budgets are summed. *ii*) The model is then trained to predict which nodes were merged at each scale. *iii*) In the *expansion* phase, merged nodes (shown in dark in the leftmost column) are expanded back (copies shown in dark), inheriting their parent’s features, budget, and connectivity. In the *refinement* phase, the model is trained to (a) identify which edges should be removed (dotted lines), (b) predict how the parent’s budget should be split across the children, and (c) refine the features of newly expanded nodes.

left side nodes being merged, and copies of the same hyperedge being subsequently collapsed. We limit node mergings to at most two per cluster, which consequently bounds hyperedge mergings to at most three (Gailhard et al., 2025). For each resulting cluster, we track two quantities. First, the *budget* is the total number of nodes contained in the cluster. Initially, each node and hyperedge has a budget of 1; when clusters merge, their budgets are summed and assigned to the new super-cluster. The node budget provides a local growth constraint, indicating how many fine-level nodes each coarse cluster should expand into and guiding refinement in regions that grow at different rates. Second, the super-cluster’s feature is computed as the weighted mean of the features of the clusters it absorbs, so it represents the average feature of all its contained nodes.

We denote node (left-side) budgets and features by $\mathbf{b}_L \in \mathbb{N}^{|\mathcal{V}_L|}$ and \mathbf{F}_L , and hyperedge (right-side) budgets and features by $\mathbf{b}_R \in \mathbb{N}^{|\mathcal{V}_R|}$ and \mathbf{F}_R . The following definition formalizes our coarsening process.

Definition 1 (Bipartite graph coarsening). Let H be an arbitrary hypergraph, $C = (\mathcal{V}_c, \mathcal{E}_c)$ its clique expansion, and $B = (\mathcal{V}_L, \mathcal{V}_R, \mathcal{E}_b, \mathbf{b}_L, \mathbf{b}_R, \mathbf{F}_L, \mathbf{F}_R)$ its *featured* bipartite representation. Let $\mathcal{P}_L = \{\mathcal{V}^{(1)}, \dots, \mathcal{V}^{(n)}\}$ be a partitioning² of the node set \mathcal{V}_L such that each set $\mathcal{V}^{(p)}$ induces a connected subgraph in C . We construct an intermediate coarsening $\tilde{B}(B, \mathcal{P}_L) = (\tilde{\mathcal{V}}_L, \mathcal{V}_R, \tilde{\mathcal{E}}_b, \bar{\mathbf{b}}_L, \mathbf{b}_R, \bar{\mathbf{F}}_L, \mathbf{F}_R)$ by merging each part $\mathcal{V}^{(p)}$ into a single node $v^{(p)} \in \tilde{\mathcal{V}}_L$, and by defining:

$$\bar{\mathbf{b}}_L[p] = \sum_{v \in \mathcal{V}^{(p)}} \mathbf{b}_L[v], \quad \bar{\mathbf{F}}_L[p] = \frac{1}{\bar{\mathbf{b}}_L[p]} \sum_{v \in \mathcal{V}^{(p)}} \mathbf{b}_L[v] \mathbf{F}_L[v], \quad (1)$$

for every merged node $\mathcal{V}^{(p)}$. An edge $e_{\{p,q\}} \in \tilde{\mathcal{E}}_b$ is added between $v^{(p)} \in \tilde{\mathcal{V}}_L$ and $v^{(q)} \in \mathcal{V}_R$ if there exists an edge $e_{\{i,q\}} \in \mathcal{E}_b$ in the original bipartite representation between some $v^{(i)} \in \mathcal{V}^{(p)}$ and $v^{(q)}$.

To complete the coarsening process, we define an equivalence relation $v_1 \sim v_2 \iff \mathcal{N}(v_1) = \mathcal{N}(v_2)$ on \mathcal{V}_R , where $\mathcal{N}(v)$ denotes the set of neighbors of v , *i.e.*, we consider right side nodes having

²That is, $\mathcal{V}^{(p)} \subseteq \mathcal{V}_L$, $\bigcup_{i=1}^n \mathcal{V}^{(i)} = \mathcal{V}_L$, and $\mathcal{V}^{(i)} \cap \mathcal{V}^{(j)} = \emptyset \forall 1 \leq i, j \leq n$.

the same set of neighbors as equivalent, or in other words we consider hyperedges containing the same set of nodes as equivalent. This induces a partitioning $\mathcal{P}_R = \{\mathcal{V}_R^{(1)}, \dots, \mathcal{V}_R^{(m)}\}$, allowing us to construct the fully coarsened bipartite representation $\bar{B}(\bar{B}, \mathcal{P}_L) = (\bar{\mathcal{V}}_L, \bar{\mathcal{V}}_R, \bar{\mathcal{E}}_b, \bar{\mathbf{b}}_L, \bar{\mathbf{b}}_R, \bar{\mathbf{F}}_L, \bar{\mathbf{F}}_R)$ by merging each part $\mathcal{V}_R^{(p)}$ into a single node $v_R^{(p)} \in \bar{\mathcal{V}}_R$, similarly to the construction of $\bar{\mathcal{V}}_L$, and by defining:

$$\bar{\mathbf{b}}_R[p] = \sum_{v \in \mathcal{V}_R^{(p)}} \mathbf{b}_R[v], \quad \bar{\mathbf{F}}_R[p] = \frac{1}{\bar{\mathbf{b}}_R[p]} \sum_{v \in \mathcal{V}_R^{(p)}} \mathbf{b}_R[v] \mathbf{F}_R[v], \quad (2)$$

for every merged hyperedge $\mathcal{V}_R^{(p)}$.

Remark 2. Informally, we cluster nodes in the clique expansion, merge the corresponding left-side nodes of the bipartite graph, and compute their budgets and features as described above. Right-side nodes (hyperedges) connected to the same left-side nodes are then merged. In our implementation, we select nodes for clustering using spectrum-preserving coarsening (Loukas, 2019). In the final reduction—when only one node and one hyperedge remain—we replace their features with zero matrices to yield a feature-agnostic initialization for generation. Please note that cluster size vectors introduced in (Bergmeister et al., 2024) correspond to the sizes of all sets $\mathcal{V}_L^{(p)}$ and $\mathcal{V}_R^{(p)}$, *i.e.*, the number of mergings for each cluster at this specific step, and not the number of nodes absorbed since the initialization of coarsening—which we call *budget*. Further details about the coarsening methodology are provided in Appendix B. Some visual examples of coarsening sequences are provided in Appendix C.

We provide the following theoretical result regarding the feature merging process in FAHNES.

Proposition 3. *Setting cluster features as the mean of the nodes they contain minimizes the mean squared error between the features of the fully expanded hypergraph and those of the original hypergraph.*

3.3 BUDGETED EXPANSION AND REFINEMENT

Once multi-scale representations for the dataset of hypergraphs are produced by the coarsening procedure, the objective is to train a model able to reverse it. In this section, we define the inverse of the coarsening procedure, which the model will learn to imitate. It comprises two stages: expansion (“upsampling” by duplicating nodes and hyperedges) and refinement (adjusting connectivity, budgets, and features). This process exclusively works on the bipartite representation.

Since our framework only conditions on the total number of nodes in the final hypergraph, we maintain a single node budget vector $\mathbf{b} \in \mathbb{N}^{|\mathcal{V}_L|}$ for the node (left-side) partition and discard the hyperedge (right-side) budget. To undo mergings, clusters are recursively split into multiple child nodes. Each child initially copies the parent’s connections and inherits its feature and budget, before a subsequent refinement step, then: *i*) redistributes the budget among the children, *ii*) updates their features, and *iii*) removes edges that should not persist at the finer scale.

Formally, the expansion and refinement steps can be described as follows.

Definition 4 (Bipartite graph expansion). Given a bipartite graph $B = (\mathcal{V}_L, \mathcal{V}_R, \mathcal{E}, \mathbf{b}, \mathbf{F}_L, \mathbf{F}_R)$ and two expansion size vectors $\mathbf{v}_L \in \mathbb{N}^{|\mathcal{V}_L|}$, $\mathbf{v}_R \in \mathbb{N}^{|\mathcal{V}_R|}$, denoting the number of duplication for nodes (left side) and hyperedges (right side), respectively. Let $\bar{B}(B, \mathbf{v}_L, \mathbf{v}_R) = (\bar{\mathcal{V}}_L, \bar{\mathcal{V}}_R, \bar{\mathcal{E}}, \mathbf{b}^{\text{expanded}}, \mathbf{F}_L^{\text{expanded}}, \mathbf{F}_R^{\text{expanded}})$ denote the expansion of B , whose node sets, budgets, and features are given by:

$$\begin{aligned} \bar{\mathcal{V}}_L &= \mathcal{V}_L^{(1)} \cup \dots \cup \mathcal{V}_L^{(|\mathcal{V}_L|)}, \text{ where } \mathcal{V}_L^{(p)} = \{v_L^{(p,i)} \mid 1 \leq i \leq \mathbf{v}_L[p]\} \text{ for } 1 \leq p \leq |\mathcal{V}_L|, \\ \bar{\mathcal{V}}_R &= \mathcal{V}_R^{(1)} \cup \dots \cup \mathcal{V}_R^{(|\mathcal{V}_R|)}, \text{ where } \mathcal{V}_R^{(p)} = \{v_R^{(p,i)} \mid 1 \leq i \leq \mathbf{v}_R[p]\} \text{ for } 1 \leq p \leq |\mathcal{V}_R|, \\ \mathbf{b}^{\text{expanded}}[p, i] &= \mathbf{b}[p] \text{ for } 1 \leq i \leq \mathbf{v}_L[p], 1 \leq p \leq |\mathcal{V}_L|, \\ \mathbf{F}_L^{\text{expanded}}[p, i] &= \mathbf{F}_L[p] \text{ for } 1 \leq i \leq \mathbf{v}_L[p], 1 \leq p \leq |\mathcal{V}_L|, \\ \mathbf{F}_R^{\text{expanded}}[p, i] &= \mathbf{F}_R[p] \text{ for } 1 \leq i \leq \mathbf{v}_R[p], 1 \leq p \leq |\mathcal{V}_R|. \end{aligned} \quad (3)$$

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

Remark 5. Expansion thus acts as a *clone-and-rewire* operation: vertices and hyperedges are duplicated, and each child inherits every incident edge of its parent.

Definition 6 (Bipartite graph refinement). Given a bipartite graph $\tilde{B} = (\tilde{\mathcal{V}}_L, \tilde{\mathcal{V}}_R, \tilde{\mathcal{E}}, \mathbf{b}, \mathbf{F}_L^{\text{expanded}}, \mathbf{F}_R^{\text{expanded}})$, an edge selection vector $\mathbf{e} \in \{0, 1\}^{|\tilde{\mathcal{E}}|}$, a budget split vector $\mathbf{f} \in [0, 1]^{|\tilde{\mathcal{V}}_L|}$, satisfying $\sum_{v \in \mathcal{V}_L^{(p)}} \mathbf{f}[v] = 1$ for all cluster $\mathcal{V}_L^{(p)}$ in $\tilde{\mathcal{V}}_L$, and two feature refinement vectors $\mathbf{F}_L^{\text{refine}}$ and $\mathbf{F}_R^{\text{refine}}$ with the same dimensions as $\mathbf{F}_L^{\text{expanded}}$ and $\mathbf{F}_R^{\text{expanded}}$, let $B(\tilde{B}, \mathbf{e}, \mathbf{f}, \mathbf{F}_L^{\text{refine}}, \mathbf{F}_R^{\text{refine}}) = (\tilde{\mathcal{V}}_L, \tilde{\mathcal{V}}_R, \mathcal{E}, [\mathbf{b} \odot \mathbf{f}], \mathbf{F}_L^{\text{refine}}, \mathbf{F}_R^{\text{refine}})$ denote the refinement of \tilde{B} , where $\mathcal{E} \subseteq \tilde{\mathcal{E}}$ such that the i -th edge $e_{(i)} \in \mathcal{E}$ if and only if $\mathbf{e}[i] = 1$.

Remark 7. Edges are selectively removed based on the binary indicator vector \mathbf{e} , and features are updated with new predictions. Node budgets are divided among child nodes according to the split proportions specified by the vector \mathbf{f} . Since budgets must be integers, the resulting values are rounded. In the case of a tie (e.g., when an odd number must be split evenly), the child with the lowest index receives the larger share, and the remaining budget is distributed among the others accordingly.

3.4 PROBABILISTIC MODELING

We now present a formalization of our learning framework, generalizing (Bergmeister et al., 2024; Gailhard et al., 2025). Let $\{H^{(1)}, \dots, H^{(N)}\}$ denote a set of *i.i.d.* hypergraph instances. Our objective is to approximate the unknown generative process by learning a distribution $p(H)$. We model the marginal likelihood of each hypergraph H as a sum over the likelihoods of its bipartite representation’s expansion sequences $p(H) = p(B) = \sum_{\varpi \in \Pi(B)} p(\varpi)$, where $\Pi(B)$ denotes the set of valid expansion sequences from a minimal bipartite graph to the full bipartite representation B corresponding to H . Each intermediate $B^{(l-1)}$ is generated by expanding and refining its predecessor, in accordance with Definitions 4 and 6.

To simplify notations, we drop the superscript for $\mathbf{F}^{\text{refine}}$ and simply write \mathbf{F} . Assuming a Markovian generative structure, the likelihood of a specific expansion sequence ϖ is factorized as:

$$p(\varpi) = \overbrace{p(B^{(L)})}^1 \prod_{l=L}^1 p(B^{(l-1)} | B^{(l)}) = \prod_{l=L}^1 p(\mathbf{e}^{(l-1)}, \mathbf{f}^{(l-1)}, \mathbf{F}_L^{(l-1)}, \mathbf{F}_R^{(l-1)} | \tilde{B}^{(l-1)}) p(\mathbf{v}_L^{(l)}, \mathbf{v}_R^{(l)} | B^{(l)}). \quad (4)$$

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

$$p(\varpi) = p(\mathbf{e}^{(0)}, \mathbf{f}^{(0)}, \mathbf{F}_L^{(0)}, \mathbf{F}_R^{(0)} | \tilde{B}^{(0)}) p(\mathbf{v}_L^{(L)}, \mathbf{v}_R^{(L)}) \left[\prod_{l=L-1}^1 p(\mathbf{v}_L^{(l)}, \mathbf{v}_R^{(l)} | B^{(l)}) p(\mathbf{e}^{(l)}, \mathbf{f}^{(l)}, \mathbf{F}_L^{(l)}, \mathbf{F}_R^{(l)} | \tilde{B}^{(l)}) \right], \quad (5)$$

where $p(\mathbf{v}_L^{(L)}, \mathbf{v}_R^{(L)}) = p(\mathbf{v}_L^{(L)}, \mathbf{v}_R^{(L)} | B^{(L)})$.

We assume that the variables $\mathbf{v}_L^{(l)}$ and $\mathbf{v}_R^{(l)}$ are conditionally independent of $\tilde{B}^{(l)}$ when conditioned on $B^{(l)}$:

$$p(\mathbf{v}_L^{(l)}, \mathbf{v}_R^{(l)} | B^{(l)}, \tilde{B}^{(l)}) = p(\mathbf{v}_L^{(l)}, \mathbf{v}_R^{(l)} | B^{(l)}). \quad (6)$$

This allows us to write the combined likelihood as:

$$p(\mathbf{v}_L^{(l)}, \mathbf{v}_R^{(l)} | B^{(l)}) p(\mathbf{e}^{(l)}, \mathbf{f}^{(l)}, \mathbf{F}_L^{(l)}, \mathbf{F}_R^{(l)} | \tilde{B}^{(l)}) = p(\mathbf{v}_L^{(l)}, \mathbf{v}_R^{(l)}, \mathbf{e}^{(l)}, \mathbf{f}^{(l)}, \mathbf{F}_L^{(l)}, \mathbf{F}_R^{(l)} | \tilde{B}^{(l)}), \quad (7)$$

which corresponds to the combined expansion and refinement step that inverts coarsening at depth l .

During training, the model approximates the right-hand side of (7), learning to generate, for each expansion and refinement step, \mathbf{v}_L , \mathbf{v}_R , \mathbf{e} , \mathbf{f} , and $\mathbf{F}_L^{\text{refine}}$, $\mathbf{F}_R^{\text{refine}}$. Once trained, the model generates a hypergraph with N nodes through successive expansion and refinement steps:

1. *Initialization.* Start from a minimal bipartite graph: $B^{(L)} = (\{1\}, \{2\}, \{(1, 2)\}, (N))$, consisting of a single node on each side connected by one edge. The left-side node is assigned the full node budget, *i.e.*, $\mathbf{b} = [N]$. If node and hyperedge features need to be generated, \mathbf{F}_L and \mathbf{F}_R are initialized as zero matrices.
2. *Expansion and refinement.* 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. *Hypergraph reconstruction.* Once the final bipartite graph is generated, construct the hypergraph by collapsing each right-side node into a hyperedge connecting its adjacent left-side nodes.

3.5 MINIBATCH OT-COUPLING

Graphs and hypergraphs lack a natural node ordering, making it challenging to align predictions with their targets. After an expansion, the model may produce a permutation of the correct predictions; yet, without alignment, the computed loss will be large despite the prediction being correct. This mismatch introduces significant noise into the learning signal. To address this, we generalize *minibatch OT-coupling* (Tong et al., 2024; Pooladian et al., 2023) from image generation to our setting. In its original form, OT-coupling aligns samples from the prior and target distributions within a minibatch via an optimal transport plan, reindexing prior samples to minimize the matching cost. Formally, given prior samples $\mathbf{X} \in \mathbb{R}^{B \times d}$ and targets $\mathbf{Y} \in \mathbb{R}^{B \times d}$, OT-coupling finds:

$$\mathbf{P}^* = \arg \min_{\mathbf{P} \in \Pi_B} \|\mathbf{XP} - \mathbf{Y}\|^2, \quad (8)$$

where \mathbf{P} is a permutation matrix and Π_B denotes the set of all $B \times B$ permutations.

In our context, for a given bipartite graph with n nodes, adjacency matrix $\mathbf{A} \in \{0, 1\}^{n \times n}$, expanded budgets $\mathbf{b} \in \mathbb{N}^n$ and expanded features $\mathbf{F} \in \mathbb{R}^{n \times d}$ where d is the dimension of node or hyperedge features, \mathbf{X} and \mathbf{Y} correspond to prior samples and targets, respectively, and \mathbf{P} is implemented as a node-swapping operation. To avoid bias, swaps are only allowed between *equivalent* nodes—those structurally and feature-wise indistinguishable up to noise. Swapping non-equivalent nodes would alter the learned distribution and degrade model performance. This leads to the following problem:

$$\mathbf{P}^* = \arg \min_{\mathbf{P} \in \Pi_n} \|\mathbf{PX} - \mathbf{Y}\|^2 \quad \text{s.t.} \quad \mathbf{P}^\top \mathbf{A} \mathbf{P} = \mathbf{A}, \mathbf{P} \mathbf{b} = \mathbf{b}, \mathbf{P} \mathbf{F} = \mathbf{F}. \quad (9)$$

To reduce overhead, we only consider permuting children from a single cluster expansion, where equivalence holds naturally. In our implementation, with exactly two or three children per cluster, only two or six permutations are possible, making the operation lightweight and easily parallelizable with standard tensor operations.

Proposition 8 (Informal). *Minibatch OT-coupling in our framework preserves the target distribution and produces shorter target paths.*

3.6 GENERALIZATION TO GRAPHS

Our framework can be directly extended to generate standard graphs by adapting the coarsen–expand procedure of Bergmeister et al. (2024). As in the hypergraph setting, each node begins with a budget of one, which is summed when nodes are merged, while features are aggregated through a weighted average with weights proportional to node budgets. During expansion, clusters propagate their budgets and features to child nodes, and the model predicts how to split the budget among them while refining their features. Graph generation, therefore, starts from a single node with a budget equal to the desired size and proceeds through successive expansion and refinement steps. To ensure stable training and alignment between predicted and target structures, minibatch OT-coupling is applied to groups of expanded nodes exactly as in the hypergraph case.

4 EXPERIMENTS AND RESULTS

In this section, we detail our experimental setup, covering datasets, evaluation metrics, implementation choices, baselines, results, and ablation studies. Full experimental details are provided in Appendix D. Visualizations of the generated samples are provided in Appendix I.

4.1 DATASETS AND EXPERIMENTAL SETUP

Datasets. We evaluate our method on five featureless datasets: Stochastic Block Model (*SBM*) (Kim et al., 2018), *Ego* (Comrie & Kleinberg, 2021), *Tree* (Nieminen & Peltola, 1999), ModelNet40 *bookshelf*, and ModelNet40 *piano* (Wu et al., 2015). We also evaluate FAHNES on two featured hypergraphs datasets, comprising two sets of 3D meshes: *Manifold40 bench* and *Manifold40 airplane* (Hu et al., 2022). Additionally, we adapt our method to graphs using the methodology outlined in Section 3.6 and evaluate it on two point cloud datasets sampled on the meshes of the two previous mesh datasets. For 3D mesh and point cloud datasets, node features are 3D positions, and edges or hyperedges do not have features.

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

Method	SBM Hypergraphs			Ego Hypergraphs			Tree Hypergraphs		
	Valid SBM \uparrow	Node Num \downarrow	Spectral \downarrow	Valid Ego \uparrow	Node Num \downarrow	Spectral \downarrow	Valid Tree \uparrow	Node Num \downarrow	Spectral \downarrow
HyperPA	2.5	0.075	0.273	0.0	35.830	0.237	0.0	2.350	0.159
VAE	0.0	0.375	0.024	0.0	47.580	0.133	0.0	9.700	0.124
GAN	0.0	1.200	0.059	0.0	60.350	0.230	0.0	6.000	0.089
Diffusion	0.0	0.150	0.031	0.0	4.475	0.190	0.0	2.225	0.127
HYGENE	65.0	0.525	0.010	90.0	12.550	0.004	77.5	0.000	0.012
FAHNES	87.8 \pm 3.1	0.029 \pm 0.009	0.006 \pm 0.004	99.5 \pm 1.1	0.128 \pm 0.171	0.004 \pm 0.003	89.7 \pm 6.0	0.000 \pm 0.000	0.003 \pm 0.002

Table 2: Evaluation on ModelNet40.

Method	ModelNet40 Bookshelf		ModelNet40 Piano	
	Node Num \downarrow	Spectral \downarrow	Node Num \downarrow	Spectral \downarrow
HyperPA	8.025	0.048	0.825	0.067
VAE	47.450	0.190	75.350	0.396
GAN	0.000	0.476	0.000	0.697
Diffusion	0.000	0.079	0.050	0.113
HYGENE	69.730	0.068	42.520	0.117
FAHNES	0.135 \pm 0.276	0.024 \pm 0.015	0.846 \pm 1.009	0.040 \pm 0.026

Table 3: Evaluation on the graph point cloud datasets.

Method	Airplane Point Clouds			Bench Point Clouds		
	Chamfer Dist \downarrow	Spectral \downarrow	Ratio \downarrow	Chamfer Dist \downarrow	Spectral \downarrow	Ratio \downarrow
DiGress	OOM	OOM	OOM	OOM	OOM	OOM
DeFoG	OOM	OOM	OOM	OOM	OOM	OOM
FAHNES	0.094 \pm 0.006	0.005 \pm 0.004	67.311 \pm 44.903	0.130 \pm 0.000	0.004 \pm 0.000	84.423 \pm 0.000

Metrics. For featureless hypergraphs, we follow the evaluation criteria in (Gailhard et al., 2025). These include: *i*) structural comparison metrics such as *Node Num* (difference in node counts); *ii*) topological analysis with *Spectral* (maximum mean discrepancy between the spectral distributions). In scenarios where datasets enforce structural constraints, we report *Valid*—the percentage of generated samples satisfying those constraints. For all metrics except *Valid*, lower values indicate improved performance, while higher values are preferable for *Valid*. In the case of 3D meshes, we use *ChamDist* (the nearest training sample Chamfer distance), which computes the Chamfer distance between point clouds sampled from a generated sample and all training samples and outputs the minimum distance. For graph point cloud datasets, we report the nearest training sample Chamfer distance, the maximum mean discrepancy (MMD) between the spectral distributions of the generated samples and training set, and the ratio of various metrics computed for the generated samples and test set (see Appendix D for details). Detailed numerical results with additional metrics can be found in Appendix H.

Implementation details. Our implementation uses a custom flow-matching framework (Lipman et al., 2022) together with a *local PPGN* architecture (Bergmeister et al., 2024). We also leverage our budget component using graph inpainting techniques, enabling the model to learn only the necessary parts of the distribution: *i*) budget splits are fixed to 1 for unexpanded clusters or those with a budget of 1, *ii*) equal splits are enforced for expanded clusters with a budget of 2, *iii*) clusters of size one are not allowed to expand, and *iv*) the features of non-expanded clusters are copied unchanged. Extensive implementation details are provided in Appendix E. The training and sampling procedures are explained in Appendix F. We also analyze the computational complexity in Appendix G.

Baselines. For the featureless hypergraphs, we compare FAHNES against HYGENE (Gailhard et al., 2025), HyperPA (Do et al., 2020), a Variational Autoencoder (VAE) (Kingma & Welling, 2013), a Generative Adversarial Network (GAN) (Goodfellow et al., 2020), and a standard 2D diffusion model (Ho et al., 2020) trained on incidence matrix images, where hyperedge membership is represented by white pixels and absence by black pixels. For 3D meshes, we compare FAHNES with a sequential disjoint generation baseline, in which the hypergraph topology is generated first, followed by the feature generation. For the graph point cloud datasets, we compare FAHNES against DeFoG (Qin et al., 2024) and DiGress (Vignac et al., 2023), two state-of-the-art flat models for graph generation.

4.2 RESULTS AND DISCUSSION

Comparison with the baselines. Tables 1 and 2 show the comparisons for the featureless hypergraphs. We see that our node-budget and OT-coupling components improve the generation quality. Regarding the Manifold40 featured hypergraph dataset³, **FAHNES obtains a Chamfer distance of 0.073 and 0.049 for bench and airplane, respectively, while the sequential baseline reaches 0.143 and 0.117, respectively.** This shows the better modeling capability of FAHNES in jointly generating topology and features in hypergraphs instead of a simple two-step approach. About the graph point cloud

³The full set of numerical results for the Manifold40 datasets is provided in Appendix H.

Table 4: Ablation studies on the node budget (Budg.) and minibatch OT-coupling (Coup.) for *SBM*, *Ego*, and *Tree* hypergraphs.

Budg.	Coup.	SBM			Tree			Ego		
		Valid \uparrow	NodeNum \downarrow	Spectral \downarrow	Valid \uparrow	NodeNum \downarrow	Spectral \downarrow	Valid \uparrow	NodeNum \downarrow	Spectral \downarrow
\checkmark	\checkmark	87.8 \pm 3.1	0.029 \pm 0.009	0.006 \pm 0.004	89.7 \pm 6.0	0.000 \pm 0.000	0.003 \pm 0.002	99.5 \pm 1.1	0.128 \pm 0.171	0.004 \pm 0.003
\times	\checkmark	85.3 \pm 5.9	0.044 \pm 0.078	0.006 \pm 0.004	90.7 \pm 6.7	0.000 \pm 0.000	0.004 \pm 0.001	100.0 \pm 0.0	0.118 \pm 0.158	0.005 \pm 0.003
\checkmark	\times	86.7 \pm 6.7	0.039 \pm 0.014	0.006 \pm 0.004	89.3 \pm 3.6	0.000 \pm 0.000	0.005 \pm 0.011	99.9 \pm 0.4	0.073 \pm 0.050	0.007 \pm 0.012
\times	\times	84.6 \pm 4.6	0.049 \pm 0.045	0.007 \pm 0.007	95.6 \pm 3.7	0.000 \pm 0.000	0.005 \pm 0.003	99.5 \pm 1.1	0.117 \pm 0.155	0.004 \pm 0.003

Table 5: Ablation studies on the node budget (Budg.) and minibatch OT-coupling (Coup.) for ModelNet and ManifoldNet datasets.

Budg.	Coup.	ModelNet Bookshelf		ModelNet Piano		ManifoldNet Airplane			ManifoldNet Bench		
		NodeNum \downarrow	Spectral \downarrow	Node Num \downarrow	Spectral \downarrow	ChamDist \downarrow	Node Num \downarrow	Spectral \downarrow	ChamDist \downarrow	NodeNum \downarrow	Spectral \downarrow
\checkmark	\checkmark	0.135 \pm 0.276	0.024 \pm 0.015	0.846 \pm 1.009	0.040 \pm 0.026	0.048 \pm 0.003	0.017 \pm 0.070	0.010 \pm 0.003	0.064 \pm 0.005	0.060 \pm 0.149	0.017 \pm 0.011
\times	\checkmark	0.940 \pm 0.917	0.032 \pm 0.013	3.622 \pm 1.822	0.055 \pm 0.040	0.079 \pm 0.019	0.426 \pm 0.820	0.014 \pm 0.007	0.090 \pm 0.003	0.240 \pm 0.265	0.018 \pm 0.009
\checkmark	\times	0.265 \pm 0.496	0.014 \pm 0.007	3.155 \pm 3.637	0.030 \pm 0.048	0.050 \pm 0.005	0.052 \pm 0.065	0.012 \pm 0.005	0.085 \pm 0.056	0.020 \pm 0.032	0.013 \pm 0.003
\times	\times	1.325 \pm 1.631	0.031 \pm 0.009	5.490 \pm 8.847	0.036 \pm 0.016	0.100 \pm 0.023	0.304 \pm 0.437	0.019 \pm 0.009	0.098 \pm 0.024	0.267 \pm 0.319	0.022 \pm 0.014

datasets, Table 3 shows that our hierarchical method is the only one able to perform at this scale, which shows its superior scalability. Detailed numerical results with more metrics are shown in Appendix H.

4.3 ABLATION STUDIES

Tables 4 and 5 present ablation studies of FAHNES for the node-budget and OT-coupling components. We observe that using budgets instead of concatenating the target size to each node embedding, like in (Bergmeister et al., 2024; Gailhard et al., 2025), improves generation quality. OT-coupling has a more nuanced effect, clearly improving quality on some datasets, like SBM or ManifoldNet Airplane, while not changing much for others. Those two components are also essential for feature generation, as lacking one of them results in much worse results, as seen by the large increase in *ChamDist* when one component is ablated.

4.4 LIMITATIONS

While our node budget mechanism helps mitigate the issue of missing nodes, it does not fully resolve it, and our method still struggles when generating very large hypergraphs, such as ModelNet Bookshelf and Piano, or very large graphs, such as point cloud datasets. These cases highlight that scalability remains challenging when both the number of nodes and the structural complexity grow substantially. Moreover, our framework currently assumes relatively simple feature distributions (e.g., 3D coordinates), and extending it to domains with richer or heterogeneous node and hyperedge attributes (such as categorical or multi-modal features) may require additional modeling components. Finally, although FAHNES improves stability through minibatch OT-coupling, training remains computationally expensive for large-scale datasets, which may limit applicability in resource-constrained settings.

5 CONCLUSION

We presented FAHNES, the first scalable hierarchical framework for jointly generating graph and hypergraph topology together with features. By integrating coarse-to-fine structural modeling with feature-aware generation, FAHNES overcomes the limitations of flat or disjoint approaches and enables the generation of complex structures at larger scales. Key innovations include a node budget mechanism, which provides fine-grained control over local growth, and minibatch OT-coupling, which improves alignment and stability during training. Extensive experiments on synthetic, 3D mesh, and point cloud datasets show that FAHNES achieves strong performance in both topology and feature generation while scaling to graph and hypergraph sizes that are out of reach for prior models. Our framework opens new directions for generative modeling of structured data, including applications in 3D geometry and circuit modeling, as well as extensions to richer feature modalities.

486 REPRODUCIBILITY STATEMENT
487

488 All proofs of propositions and theorems, and all details regarding the experiments, parameters, and
489 datasets can be found in the appendix. The code will be made public upon acceptance of the paper.
490

491 REFERENCES
492

- 493 Sinan G Aksoy, Cliff Joslyn, Carlos Ortiz Marrero, Brenda Praggastis, and Emilie Purvine. Hypernet-
494 work science via high-order hypergraph walks. *EPJ Data Science*, 2020.
495
- 496 Melih Barsbey, Rubén Ballester, Andac Demir, Carles Casacuberta, Pablo Hernández-García, David
497 Pujol-Perich, Sarp Yurtseven, Sergio Escalera, Claudio Battiloro, Mustafa Hajj, and Tolga Birdal.
498 Higher-order molecular learning: The cellular transformer. In *ICLR 2025 Workshop on Generative
499 and Experimental Perspectives for Biomolecular Design*, 2025.
- 500 Andreas Bergmeister, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Efficient
501 and scalable graph generation through iterative local expansion. In *International Conference on
502 Learning Representations*, 2024.
503
- 504 Xiaohui Chen, Jiaying He, Xu Han, and Li-Ping Liu. Efficient and degree-guided graph generation
505 via discrete diffusion modeling. In *International Conference on Machine Learning*, 2023.
- 506 Cazamere Comrie and Jon Kleinberg. Hypergraph ego-networks and their temporal evolution. In
507 *IEEE International Conference on Data Mining*, 2021.
508
- 509 Manh Tuan Do, Se-eun Yoon, Bryan Hooi, and Kijung Shin. Structural patterns and generative
510 models of real-world hypergraphs. In *ACM SIGKDD International Conference on Knowledge
511 Discovery & Data Mining*, 2020.
- 512 John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the
513 ℓ_1 -ball for learning in high dimensions. *International Conference on Machine Learning*, 2008.
514
- 515 Ian Dunn and David Ryan Koes. Mixed continuous and categorical flow matching for 3D de novo
516 molecule generation. *ArXiv*, 2024.
517
- 518 Floor Eijkelboom, Grigory Bartosh, Christian Andersson Naesseth, Max Welling, and Jan-Willem
519 van de Meent. Variational flow matching for graph generation. In *Advances in Neural Information
520 Processing Systems*, 2024.
- 521 Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam
522 Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English,
523 Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow trans-
524 formers for high-resolution image synthesis. In *International Conference on Machine Learning*,
525 2024.
- 526 Dorian Gailhard, Enzo Tartaglione, Lirida Naviner, and Jhony H Giraldo. HYGENE: A diffusion-
527 based hypergraph generation method. In *AAAI Conference on Artificial Intelligence*, 2025.
528
- 529 Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair,
530 Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the
531 ACM*, 2020.
532
- 533 Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in
534 Neural Information Processing Systems*, 2020.
- 535 Shi-Min Hu, Zheng-Ning Liu, Meng-Hao Guo, Jun-Xiong Cai, Jiahui Huang, Tai-Jiang Mu, and
536 Ralph R Martin. Subdivision-based mesh convolution networks. *ACM Transactions on Graphics*,
537 2022.
538
- 539 Hiroshi Kajino. Molecular hypergraph grammar with its application to molecular optimization. In
International Conference on Machine Learning, 2019.

- 540 Chiheon Kim, Afonso S Bandeira, and Michel X Goemans. Stochastic block model for hypergraphs:
541 Statistical limits and a semidefinite programming approach. *ArXiv*, 2018.
- 542
543 Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference*
544 *on Learning Representations*, 2013.
- 545 Derek Lim, Joshua Robinson, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Ste-
546 fanie Jegelka. Sign and basis invariant networks for spectral graph representation learning. In
547 *International Conference on Learning Representations*, 2022.
- 548 Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching
549 for generative modeling. In *International Conference on Learning Representations*, 2022.
- 550
551 Andreas Loukas. Graph reduction with spectral and cut guarantees. *Journal of Machine Learning*
552 *Research*, 2019.
- 553 Zhishang Luo, Truong Son Hy, Puoya Tabaghi, Michaël Defferrard, Elahe Rezaei, Ryan M Carey,
554 Rhett Davis, Rajeev Jain, and Yusu Wang. De-hnn: An effective neural model for circuit netlist
555 representation. In *International Conference on Artificial Intelligence and Statistics*, 2024.
- 556
557 Karolis Martinkus, Andreas Loukas, Nathanaël Perraudin, and Roger Wattenhofer. Spectre: Spectral
558 conditioning helps to overcome the expressivity limits of one-shot graph generators. In
559 *International Conference on Machine Learning*, 2022.
- 560
561 Charlie Nash, Yaroslav Ganin, SM Ali Eslami, and Peter Battaglia. PolyGen: An autoregressive
562 generative model of 3D meshes. In *International Conference on Machine Learning*, 2020.
- 563
564 J Nieminen and M Peltola. Hypertrees. *Applied mathematics letters*, 1999.
- 565
566 Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. Permu-
567 tation invariant graph generation via score-based generative modeling. In *International Conference*
568 *on Artificial Intelligence and Statistics*, 2020.
- 569
570 Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual
571 reasoning with a general conditioning layer. In *AAAI Conference on Artificial Intelligence*, 2018.
- 572
573 Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman,
574 and Ricky T. Q. Chen. Multisample flow matching: Straightening flows with minibatch couplings.
575 In *International Conference on Machine Learning*, 2023.
- 576
577 Yiming Qin, Manuel Madeira, Dorina Thanou, and Pascal Frossard. DeFoG: Discrete flow matching
578 for graph generation. *ArXiv*, 2024.
- 579
580 Ahsanur Rahman, Christopher L Poirel, David J Badger, and TM Murali. Reverse engineering
581 molecular hypergraphs. In *ACM Conference on Bioinformatics, Computational Biology and*
582 *Biomedicine*, 2012.
- 583
584 Sucheng Ren, Qihang Yu, Ju He, Xiaohui Shen, Alan Yuille, and Liang-Chieh Chen. FlowAR:
585 Scale-wise autoregressive image generation meets flow matching. In *International Conference on*
586 *Machine Learning*, 2024.
- 587
588 Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav
589 Rosov, Angela Dai, and Matthias Nießner. MeshGPT: Generating triangle meshes with decoder-
590 only transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.
- 591
592 Martin Simonovsky and Nikos Komodakis. GraphVAE: Towards generation of small graphs using
593 variational autoencoders. In *International Conference on Artificial Neural Networks*, 2018.
- 594
595 NV Starostin and VV Balashov. The use of hypergraphs for solving the problem of orthogonal
596 routing of large-scale integrated circuits with an irregular structure. *Journal of Communications*
597 *Technology and Electronics*, 2008.
- 598
599 Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling:
600 Scalable image generation via next-scale prediction. In *Advances in Neural Information Processing*
601 *Systems*, 2024.

594 Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-
595 Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models
596 with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024.
597

598 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz
599 Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information
600 Processing Systems*, 2017.

601 Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard.
602 DiGress: Discrete denoising diffusion for graph generation. In *International Conference on
603 Learning Representations*, 2023.

604 Nisheeth K. Vishnoi. $Lx = b$. *Foundations and Trends® in Theoretical Computer Science*, 2013.
605 ISSN 1551-305X.
606

607 Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong
608 Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE/CVF Conference on
609 Computer Vision and Pattern Recognition*, 2015.

610 Zhe Xu, Ruizhong Qiu, Yuzhong Chen, Huiyuan Chen, Xiran Fan, Menghai Pan, Zhichen Zeng, Ma-
611 hashweta Das, and Hanghang Tong. Discrete-state continuous-time diffusion for graph generation.
612 In *Advances in Neural Information Processing Systems*, 2024.
613

614 Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. GraphRNN: Generating
615 realistic graphs with deep auto-regressive models. In *International Conference on Machine
616 Learning*, 2018.

617 Yanqiao Zhu, Yuanqi Du, Yinkai Wang, Yichen Xu, Jieyu Zhang, Qiang Liu, and Shu Wu. A survey
618 on deep graph generation: Methods and applications. In *Learning on Graphs Conference*, 2022.
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647

This supplementary material offers additional technical details and formal proofs to complement the main paper. The document is structured as follows: Formal proofs of the main propositions are presented in Appendix A. Appendix B describes our procedure for sampling coarsening sequences. Appendix C provides illustrative examples of coarsening sequences. Appendix D outlines the experimental setup, including hyperparameter choices and numerical results. Appendix E discusses more detailed implementation details of FAHNES. Algorithms for model training and sampling featured hypergraphs are detailed in Appendix F. Appendix G analyzes the algorithmic complexity of our approach. Appendix H presents detailed numerical results for the ablation studies. Appendix I presents visual comparisons between training and generated samples.

A PROOFS

A.1 AVERAGING NODE FEATURES FOR CLUSTERS' FEATURES

Proposition 9. *Setting cluster features as the average of the nodes they contain minimizes the Mean Squared Error (MSE) between the fully expanded hypergraph and the original hypergraph.*

Proof. Let $H = (\mathcal{V}, \mathcal{E})$ be a hypergraph with node set \mathcal{V} and hyperedge set \mathcal{E} , and let $H_l = (\mathcal{V}_l, \mathcal{E}_l)$ denote the lifted hypergraph obtained by expanding each cluster $\mathcal{C} \subseteq \mathcal{V}$ of size $|\mathcal{C}|$ -clique. By construction, there exists a bijection $\phi : \mathcal{V} \rightarrow \mathcal{V}_l$ mapping each original node to its corresponding lifted node. Suppose each node $v \in \mathcal{V}$ is associated with a feature vector $\mathbf{x}_v \in \mathbb{R}^d$, and each cluster $\mathcal{C} \subseteq \mathcal{V}$ is assigned a cluster feature vector $\mathbf{x}_{\mathcal{C}} \in \mathbb{R}^d$, which is inherited by all lifted nodes $\phi(v)$ for $v \in \mathcal{C}$. Define the mean squared error between the original node features and the cluster features in the lifted hypergraph as:

$$\text{MSE} = \sum_{v \in \mathcal{V}} \|\mathbf{x}_v - \mathbf{x}_{C(v)}\|^2, \quad (10)$$

where $C(v)$ denotes the cluster containing node v .

To minimize the MSE, it suffices to minimize, for each cluster \mathcal{C} ,

$$J_{\mathcal{C}}(\mathbf{x}_{\mathcal{C}}) = \sum_{v \in \mathcal{C}} \|\mathbf{x}_v - \mathbf{x}_{\mathcal{C}}\|^2. \quad (11)$$

Since $J_{\mathcal{C}}$ is a convex quadratic function in $\mathbf{x}_{\mathcal{C}}$, we find its minimum by setting the gradient to zero:

$$\nabla_{\mathbf{x}_{\mathcal{C}}} J_{\mathcal{C}}(\mathbf{x}_{\mathcal{C}}) = \sum_{v \in \mathcal{C}} 2(\mathbf{x}_{\mathcal{C}} - \mathbf{x}_v) = 2|\mathcal{C}|\mathbf{x}_{\mathcal{C}} - 2 \sum_{v \in \mathcal{C}} \mathbf{x}_v = \mathbf{0}. \quad (12)$$

Solving for $\mathbf{x}_{\mathcal{C}}$, we obtain

$$\mathbf{x}_{\mathcal{C}} = \frac{1}{|\mathcal{C}|} \sum_{v \in \mathcal{C}} \mathbf{x}_v, \quad (13)$$

which is the arithmetic mean of the feature vectors in the cluster \mathcal{C} . \square

A.2 MINIBATCH OT-COUPLING

Let B be a bipartite graph. Let \mathbf{b} be its current budget repartition, \mathbf{F}_L its node features, and \mathbf{F}_R its hyperedge features. Denote $\mathbf{x} = (\mathbf{v}_L, \mathbf{v}_R, \mathbf{e}, \mathbf{f}, \mathbf{F}_L^{\text{refine}}, \mathbf{F}_R^{\text{refine}})$, *i.e.*, the predictions of the model. In the following, all distributions are conditioned on \mathbf{b} , \mathbf{F}_L and \mathbf{F}_R .

Proposition 10 (Marginal preservation). *Under the OT-coupling of Algorithm 2, the joint distribution:*

$$q(\mathbf{x}_0, \mathbf{x}_1)$$

has marginals:

$$q_0(\mathbf{x}_0), \quad q_1(\mathbf{x}_1).$$

Definition 11 (Isomorphism of bipartite graphs in our setting). Let

$$B_1 = (\mathcal{V}_L^1, \mathcal{V}_R^1, \mathcal{E}^1, \mathbf{b}^1, \mathbf{F}_L^1, \mathbf{F}_R^1), \quad B_2 = (\mathcal{V}_L^2, \mathcal{V}_R^2, \mathcal{E}^2, \mathbf{b}^2, \mathbf{F}_L^2, \mathbf{F}_R^2),$$

be bipartite graphs.

We say $B_1 \cong B_2$ (are isomorphic in our setting) if there exist bijections

$$\sigma_L : \mathcal{V}_L^1 \rightarrow \mathcal{V}_L^2, \quad \sigma_R : \mathcal{V}_R^1 \rightarrow \mathcal{V}_R^2$$

such that:

1. Edge structure is preserved: $(v, w) \in \mathcal{E}^1 \iff (\sigma_L(v), \sigma_R(w)) \in \mathcal{E}^2$,
2. Budgets are preserved: $\mathbf{b}^1(v) = \mathbf{b}^2(\sigma_L(v))$ for all $v \in \mathcal{V}_L^1$, $\mathbf{b}^1(w) = \mathbf{b}^2(\sigma_R(w))$ for all $w \in \mathcal{V}_R^1$,
3. Node and hyperedge features are preserved: $\mathbf{F}_L^1(v) = \mathbf{F}_L^2(\sigma_L(v))$, $\mathbf{F}_R^1(w) = \mathbf{F}_R^2(\sigma_R(w))$.

Proof. Let f be an arbitrary test function defined on bipartite graphs. Algorithm 2 swaps noise samples between nodes that are equivalent in the sense that their target graphs (conditioned on the same topology and conditioning features) remain isomorphic after swapping. That is, if \mathbf{x}_0 and \mathbf{x}'_0 differ only by such a swap, then the resulting graphs are isomorphic: $B(\mathbf{x}_0) \cong B(\mathbf{x}'_0)$. Let σ be the bijective reindexing function corresponding to this isomorphism. Since f is defined on graphs and graphs are invariant under isomorphism, we have:

$$f(\mathbf{x}_0) = f(\sigma(\mathbf{x}_0)). \quad (14)$$

Therefore:

$$\mathbb{E}_{q(\mathbf{x}_0, \mathbf{x}_1)}[f(\mathbf{x}_0)] = \mathbb{E}_{q(\mathbf{x}_1)}[\mathbb{E}_{q(\mathbf{x}_0|\mathbf{x}_1)}[f(\mathbf{x}_0)]] \quad (15)$$

$$= \mathbb{E}_{q(\mathbf{x}_1)}[\mathbb{E}_{q(\mathbf{x}_0)}[f(\sigma(\mathbf{x}_0))]] \quad (16)$$

$$= \mathbb{E}_{q(\mathbf{x}_1)}[\mathbb{E}_{q(\mathbf{x}_0)}[f(\mathbf{x}_0)]] \quad (17)$$

$$= \mathbb{E}_{q(\mathbf{x}_0)}[f(\mathbf{x}_0)]. \quad (18)$$

where 16 comes from the transfer formula.

Thus, the marginal distribution over \mathbf{x}_0 remains unchanged. The same argument applies symmetrically for \mathbf{x}_1 by using σ^{-1} , concluding the proof. \square

Proposition 12 (Shorter paths). *Under the OT-coupling of Algorithm 2, the loss-induced paths for a given minibatch are, on average, shorter than without OT-coupling. Formally, let π be the permutation returned by Algorithm 2 for starting points \mathbf{x}^0 and target points \mathbf{x}^1 . Then*

$$\sum_i \|\mathbf{x}_i^0 - \mathbf{x}_i^1\|^2 \geq \sum_i \|\mathbf{x}_{\pi(i)}^0 - \mathbf{x}_i^1\|^2. \quad (19)$$

Proof. We can rewrite the left-hand side of (19) as a sum over clusters:

$$\sum_i \|\mathbf{x}_i^0 - \mathbf{x}_i^1\|^2 = \sum_{\text{clusters } \mathcal{C}} \sum_{i \in \mathcal{C}} \|\mathbf{x}_i^0 - \mathbf{x}_i^1\|^2. \quad (20)$$

By construction, Algorithm 2 minimizes $\sum_{i \in \mathcal{C}} \|\mathbf{x}_i^0 - \mathbf{x}_i^1\|^2$ within each cluster \mathcal{C} .

Therefore,

$$\sum_{\text{clusters } \mathcal{C}} \sum_{i \in \mathcal{C}} \|\mathbf{x}_i^0 - \mathbf{x}_i^1\|^2 \geq \sum_{\text{clusters } \mathcal{C}} \sum_{i \in \mathcal{C}} \|\mathbf{x}_{\pi(i)}^0 - \mathbf{x}_i^1\|^2 \quad (21)$$

$$= \sum_i \|\mathbf{x}_{\pi(i)}^0 - \mathbf{x}_i^1\|^2, \quad (22)$$

which proves the claim. \square

B COARSENING SEQUENCE SAMPLING

This section outlines our methodology for sampling a coarsening sequence $\pi \in \Pi_F(H)$ for a given hypergraph H . The full procedure is detailed in Algorithm 1. At each coarsening step l , let $H^{(l)}$ denote the current hypergraph, $B^{(l)}$ its bipartite representation, and $C^{(l)}$ its weighted clique expansion. We begin by sampling a target reduction fraction $\text{red_frac} \sim \mathcal{U}([\rho_{\min}, \rho_{\max}])$. We then evaluate all possible contraction sets $F(C^{(l-1)})$ using a cost function c , where lower cost indicates higher preference. We employ a greedy randomized strategy that processes contraction sets in order of increasing cost. For each set:

- The set is stochastically rejected with probability $1 - \lambda$.

- If not rejected:
 - **Overlap check:** If the contraction set overlaps with any previously accepted contraction, it is discarded.
 - **Coarsening attempt:** Otherwise, we compute tentative coarsened representations C_{temp} and B_{temp} .
 - **Cluster constraint check:** If all right-side clusters in B_{temp} contain at most three nodes, the contraction is accepted.
 - **Update step:** When a contraction is accepted, we:
 - * Sum the budgets of the nodes in the contraction set to define the new cluster budget.
 - * Compute the new cluster’s node features as a weighted average of the original features, using node budgets as weights.

The loop terminates once the number of remaining nodes satisfies the stopping condition:

$$|\mathcal{V}_L^{(l-1)}| - |\bar{\mathcal{V}}_L^{(l)}| > \text{red_frac} \cdot |\mathcal{V}_L^{(l-1)}|,$$

i.e., when the number of nodes on the left side (corresponding to the original hypergraph nodes) has been reduced by the sampled fraction. This framework is flexible, allowing a variety of cost functions c , contraction families F , reduction fraction ranges $[\rho_{\min}, \rho_{\max}]$, and randomization parameters λ .

Practical considerations. To avoid oversampling overly small graphs during training, we follow the heuristic of Bergmeister et al. (2024): when the current graph has fewer than 16 nodes, we fix the reduction fraction to $\rho = \rho_{\max}$. Due to the constraint that no right-side cluster in $B^{(l)}$ may contain more than three nodes, achieving the target reduction fraction is not always feasible. However, we observe empirically that this rarely poses a problem when ρ_{\max} is reasonably small.

During training, we sample a coarsening sequence for each dataset graph, but only retain a randomly selected intermediate graph from the sequence. Thus, our practical implementation of Algorithm 1 is designed to return a single coarsened graph with associated features and budgets, rather than the full sequence π .

To improve efficiency, we incorporate the caching mechanism introduced in Bergmeister et al. (2024). Once a coarsening sequence is generated, its levels are cached. During training, a random level is selected, returned, and then removed from the cache. A new sequence is generated only when the cache for a particular graph is depleted, avoiding unnecessary recomputation.

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: $[\rho_{\min}, \rho_{\max}] = [0.1, 0.3]$.

810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863

Algorithm 1 Hypergraph coarsening sequence sampling: Randomized iterative coarsening of a hypergraph. At each step, contraction sets are selected based on cost, while ensuring right-side clusters never merge more than three at a time. Accepted contractions update the hypergraph structure, node budgets, and features.

Parameters: contraction family F , cost function c , reduction fraction range $[\rho_{\min}, \rho_{\max}]$, randomization parameter λ

Input: hypergraph H with n nodes and m hyperedges; node features \mathbf{F}_L ; hyperedge features \mathbf{F}_R

Output: coarsening sequence $\pi = (H^{(0)}, \dots, H^{(L)}) \in \Pi_F(H)$

```

1: function HYPERGRAPHCOARSENINGSEQ( $H$ )
2:    $H^{(0)} \leftarrow H$ 
3:    $B^{(0)} \leftarrow \text{BipartiteRepresentation}(H^{(0)})$ 
4:    $C^{(0)} \leftarrow \text{WeightedCliqueExpansion}(H^{(0)})$ 
5:    $\mathbf{b}_L^{(0)} \leftarrow (1, \dots, 1) \in \mathbb{R}^n$  ▷ Initial node budgets
6:    $\mathbf{b}_R^{(0)} \leftarrow (1, \dots, 1) \in \mathbb{R}^m$  ▷ Initial hyperedge budgets
7:    $\pi \leftarrow (B^{(0)}, \mathbf{b}_L^{(0)}, \mathbf{F}_L, \mathbf{F}_R)$ 
8:    $l \leftarrow 0$ 
9:   while  $|\mathcal{V}_L^{(l)}| > 1$  do
10:     $l \leftarrow l + 1$ 
11:     $\text{red\_frac} \sim \text{Uniform}([\rho_{\min}, \rho_{\max}])$  ▷ Sample reduction fraction
12:     $f \leftarrow c(\cdot, C^{(l-1)}, (\mathcal{P}^{(l-1)}, \dots, \mathcal{P}^{(0)}))$  ▷ Cost function
13:     $\text{accepted\_contractions} \leftarrow \emptyset$ 
14:    for  $S \in \text{SortedByCost}(F(C^{(l-1)}))$  do
15:      if  $\text{Random}() > \lambda$  then
16:        if  $S \cap (\bigcup_{P \in \text{accepted\_contractions}} P) = \emptyset$  then
17:           $C_{\text{temp}} \leftarrow \text{CoarsenCliqueExpansion}(C^{(l-1)}, S)$ 
18:           $B_{\text{temp}} \leftarrow \text{CoarsenBipartite}(B^{(l-1)}, S)$ 
19:          if  $\forall$  right cluster  $R \in B_{\text{temp}} : |R| \leq 3$  then
20:             $\text{accepted\_contractions} \leftarrow \text{accepted\_contractions} \cup \{S\}$ 
21:             $C^{(l)} \leftarrow C_{\text{temp}}, B^{(l)} \leftarrow B_{\text{temp}}$ 
▷ Update budgets and features for the new cluster
22:            Let  $S = \{v_1, \dots, v_k\}$ , and the new node be  $v^*$ 
23:             $\mathbf{b}_L^{(l)}[v^*] \leftarrow \sum_{i=1}^k \mathbf{b}_L^{(l-1)}[v_i]$ 
24:             $\mathbf{F}_L^{(l)}[v^*] \leftarrow \frac{1}{\mathbf{b}_L^{(l)}[v^*]} \sum_{i=1}^k \mathbf{b}_L^{(l-1)}[v_i] \cdot \mathbf{F}_L^{(l-1)}[v_i]$ 
25:          end if
26:        end if
27:      end if
28:      if  $|\mathcal{V}_L^{(l-1)}| - |\bar{\mathcal{V}}_L^{(l)}| > \text{red\_frac} \cdot |\mathcal{V}_L^{(l-1)}|$  then
29:        break
30:      end if
31:    end for
32:     $\pi \leftarrow \pi \cup \{B^{(l)}, \mathbf{b}_L^{(l)}, \mathbf{F}_L^{(l)}\}$ 
33:  end while
34:  return  $\pi$ 
35: end function

```

C EXAMPLES OF COARSENING SEQUENCES

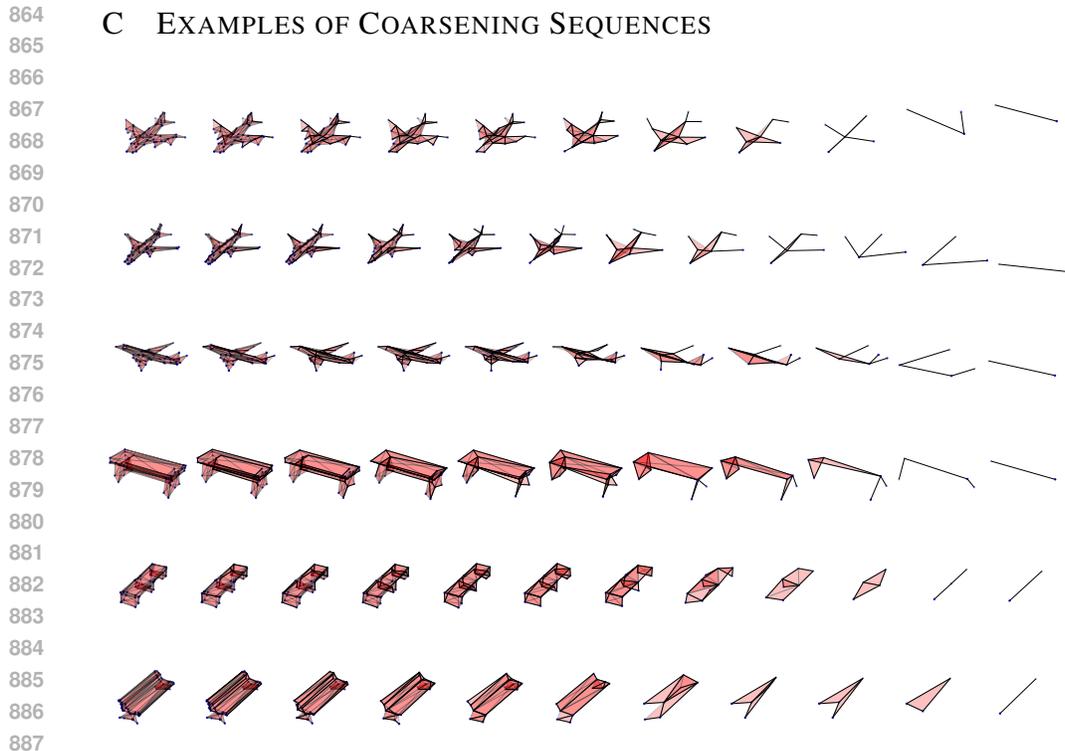


Figure 3: Examples of coarsening sequence for various meshes. Thick lines represent 2-edges.

D EXPERIMENTAL DETAILS

In this section, we detail all three types of experiments – unfeatured hypergraphs, 3D meshes and graph point clouds – individually, detailing their datasets, baselines, metrics and specific hyperparameters.

For all experiments, we use embeddings with 32 dimensions for edge selection vectors and node and hyperedge expansion numbers. When they exist, features are embedded with 128 dimensions. SignNet always has 5 layers and a hidden dimension of 128. Positional encodings always have 32 dimensions. We always use 10 layers of Local PPGN. We always use 25 sampling steps. All experiments are run for 1M steps on a single L40S. We use 8 CPU workers.

D.1 UNFEATURED HYPERGRAPHS

Datasets. Our experiments utilize five datasets: three synthetic and two real-world, consistent with those described in Gailhard et al. (2025):

- **Stochastic Block Model (SBM) hypergraphs** Kim et al. (2018): Constructed with 32 nodes split evenly into two groups. Every hyperedge connects three nodes. Hyperedges are sampled with probability 0.05 within groups and 0.001 between groups.
- **Ego hypergraphs** Comrie & Kleinberg (2021): Created by generating an initial hypergraph of 150–200 nodes with 3000 randomly sampled hyperedges (up to 5 nodes each), then extracting an ego-centric subgraph by selecting a node and retaining only hyperedges that include it.
- **Tree-structured hypergraphs** Nieminen & Peltola (1999): A tree with 32 nodes is generated using *networkx*, followed by grouping adjacent tree edges into hyperedges. Each hyperedge contains up to 5 nodes.
- **ModelNet40 meshes** Wu et al. (2015): Hypergraphs are derived from mesh topologies of selected ModelNet40 categories. To simplify computation, meshes are downsampled to fewer than 1000 vertices by iteratively merging nearby vertices. Duplicate triangles are removed, and the resulting low-poly mesh is converted into a hypergraph. We focus on the *bookshelf* and *piano* categories.

All datasets are divided into 128 training, 32 validation, and 40 testing hypergraphs.

Evaluation Metrics. We evaluate generated hypergraphs using the same suite of metrics as Gailhard et al. (2025):

- **NodeNumDiff:** Average absolute difference in node count between generated and reference hypergraphs.
- **NodeDegreeDistrWasserstein:** Wasserstein distance between node degree distributions of generated and reference hypergraphs.
- **EdgeSizeDistrWasserstein:** Wasserstein distance between hyperedge size distributions.
- **Spectral:** Maximum Mean Discrepancy (MMD) between Laplacian spectra.
- **Uniqueness:** Fraction of generated hypergraphs that are non-isomorphic to one another.
- **Novelty:** Fraction of generated hypergraphs that are non-isomorphic to training samples.
- **CentralityCloseness, CentralityBetweenness, CentralityHarmonic:** Wasserstein distances computed between centrality distributions (on edges for $s = 1$). For details see Aksoy et al. (2020).
- **ValidEgo:** For the *hypergraphEgo* dataset only, proportion of generated hypergraphs that contain a central node shared by all hyperedges.
- **ValidSBM:** For the *hypergraphSBM* dataset only, proportion of generated graphs that retain the original intra- and inter-group connectivity patterns.
- **ValidTree:** For the *hypergraphTree* dataset only, proportion of generated samples that preserve tree structure.

Baselines. We compare our method against the following baselines:

- **HyperPA** Do et al. (2020): A heuristic approach for hypergraph generation.
- **Image-based models:** We design three baseline models—Diffusion, GAN, and VAE—that operate on incidence matrix representations of hypergraphs:
 - Each model is trained to produce binary images where white pixels signify node-hyperedge membership.
 - To normalize input sizes, incidence matrices are permuted randomly and padded with black pixels.
 - Generated images are thresholded to obtain binary incidence matrices.
- **HYGENE** Gailhard et al. (2025): A hierarchical diffusion-based generator using reduction, expansion, and refinement steps.

Specific hyperparameters. We use $\lambda = 0.3$, our Local PPGN layers have a dimension of 128, and the hidden dimension for our MLP is 256. We use perturbed hypergraph expansion with a radius of 2 and dropout of 0.5. Our model has 4M parameters.

D.2 3D MESHES

Datasets. Datasets for meshes are taken from Manifold40 Hu et al. (2022), which is a reworked version of ModelNet40 Wu et al. (2015) to obtain manifold and watertight meshes. Meshes are subsequently coarsened to obtain low-poly versions of 50 vertices and 100 triangles. We use two classes:

- *Airplane* comprising 682 training samples, 21 validation samples, and 23 testing samples.
- *Bench* comprising 144 training samples, 19 validation samples, and 30 testing samples.

Metrics. We use the same metrics as for unfeatured hypergraphs. To this we add a metric *ChamDist* which computes the minimal Chamfer distance between a point cloud sampled from the surface of the generated mesh and equivalent point clouds sampled from all validation/test set meshes.

Baselines. We compare against a simple sequential baseline:

1. Our model (4M parameters) is trained for 1M steps on the topology of meshes **without** learning to generate the features.

- 972 2. A simple Local PPGN model (4M parameters) is trained for 20 epochs as a flow-matching model
 973 to learn to generate the 3D positions, with the topology fixed.
 974 3. We use the best checkpoint of the first model to generate the topology, then apply the second
 975 model on this topology to generate the 3D positions.
 976

977 **Specific hyperparameters.** We use $\lambda = 0.1$, our Local PPGN layers have a dimension of 200, and
 978 the hidden dimension for our MLP is 300. We use perturbed hypergraph expansion with a radius of 2
 979 and dropout of 0.5. Our model has 6M parameters.

980 D.3 GRAPH POINT CLOUDS

982 **Datasets.** We reuse the same datasets as for 3D meshes. Point clouds comprising 1024 nodes are
 983 sampled on the surface of each mesh, then each node is connected to its 3 nearest neighbors. Only
 984 the largest connected component is kept.

985 **Metrics. Evaluation Metrics.** We adopt the same evaluation protocol as Martinkus et al. (2022) to
 986 assess the quality of generated graphs, reporting the following metrics:
 987

- 988 • **NumDiff:** MMD between node counts distributions of generated and reference graphs.
- 989 • **Deg:** MMD between degree distributions of generated and reference graphs.
- 990 • **Clustering:** MMD between clustering coefficient distributions.
- 991 • **Orbit:** MMD between graphlet orbit count distributions.
- 992 • **Spectral:** MMD between Laplacian spectra.
- 993 • **Wavelet:** MMD between wavelet coefficient distributions.
- 994 • **Ratio:** Average ratio of generated-to-reference values across the above metrics, used as a global
 995 indicator of statistical similarity.
 996

997 To this we add a metric *ChamDist* which computes the minimal Chamfer distance between the
 998 generated mesh and all validation/test set point clouds.

999 **Baselines.** We compare against two state-of-the-art flat generative models: Defog Qin et al. (2024),
 1000 DiGress Vignac et al. (2023). As these two methods are tailored for discrete data, we use a mixed-
 1001 discrete continuous framework for continuous features.

1002 **Specific hyperparameters.** We use $\lambda = 0.3$, our Local PPGN layers have a dimension of 128, and
 1003 the hidden dimension for our MLP is 256. We use perturbed hypergraph expansion with a radius of 2
 1004 and dropout of 0.5. Our model has 4M parameters.
 1005

1006 E IMPLEMENTATION DETAILS

1007 E.1 FLOW-MATCHING FRAMEWORK

1008 We employ a flow-matching generative modeling framework Lipman et al. (2022), with endpoint
 1009 parameterization following Dunn & Koes (2024), equivalent to denoising diffusion models Ho et al.
 1010 (2020) using linear interpolation between the prior p_0 and target p_1 distributions. The goal is to align
 1011 two distributions by transporting samples from p_0 to p_1 through a learned time-dependent vector field
 1012 $\mathbf{f}(\mathbf{x}, t)$, governed by the ODE:
 1013

$$1014 \frac{\partial \mathbf{x}}{\partial t} = \mathbf{f}(\mathbf{x}, t), \quad \mathbf{x}_0 \sim p_0. \quad (23)$$

1015 Here, $\mathbf{f}(\mathbf{x}, t)$ is trained such that integrating this ODE produces samples from p_1 .
 1016

1017 **Endpoint parameterization.** Instead of directly modeling the flow field, we learn the terminal point
 1018 $\hat{\mathbf{x}}_1(\mathbf{x}_t, 1)$ of the trajectory. The flow can be recovered using:
 1019

$$1020 \mathbf{f}(\mathbf{x}_t, t) = \frac{\hat{\mathbf{x}}_1(\mathbf{x}_t, 1) - \mathbf{x}_t}{1 - t}. \quad (24)$$

1021 **Training objective.** The model is trained to minimize the expected squared error between the
 1022 predicted and true endpoints:
 1023

$$1024 \mathcal{L} = \mathbb{E}_{t, \mathbf{x}_0 \sim p_0, \mathbf{x}_1 \sim p_1} \left[\|\hat{\mathbf{x}}_1(\mathbf{x}_t, 1) - \mathbf{x}_1\|^2 \right], \quad (25)$$

where $\mathbf{x}_t = (1 - t)\mathbf{x}_0 + t\mathbf{x}_1$ is a linear interpolation between samples from the prior and target distributions. Following Esser et al. (2024), during training t is sampled using a log-normal distribution, which has been shown to improve performance.

Prior distributions. We use different prior distributions depending on the task:

- **Node and edge predictions:** Prior samples p_0 are drawn from a Gaussian distribution. Targets are either -1 or 1 , indicating binary decisions (*e.g.*, whether a node is expanded or an edge is retained).
- **Hyperedge expansion:** Similar to node and edge predictions, with targets $-1, 0$, or 1 , encoding the number of expansions (none, one, or two).
- **Budget fractions:** Prior samples p_0 are drawn from a Dirichlet distribution with concentration parameter $\alpha = 1.5$, linearly mapped to $[-1, 1]$ via $2x - 1$. The target is the budget fraction of each child node, linearly mapped to $[-1, 1]$ via $2x - 1$. If a cluster is not expanded, the corresponding budget becomes 1. Parent cluster budgets are encoded using sinusoidal positional encodings Vaswani et al. (2017) (dimension 32, base frequency 10^{-4}).
- **Feature generation:** Following Ren et al. (2024), we draw prior features from a Gaussian and predict true node features, conditioned on the parent node’s feature using a FiLM layer Perez et al. (2018).

Simplex projection via Von Neumann method. When modeling budget fractions, predictions must lie on the probability simplex. To ensure this, we project the model’s outputs using the Von Neumann projection Duchi et al. (2008), which finds the closest point (in Euclidean distance) on the simplex:

$$\Delta^K = \left\{ \mathbf{x} \in \mathbb{R}^K \mid x_i \geq 0, \sum_{i=1}^K x_i = 1 \right\}. \quad (26)$$

- i)* Sort $\mathbf{z} \in \mathbb{R}^K$ into a descending vector \mathbf{u} , such that $u_1 \geq u_2 \geq \dots \geq u_K$.
- ii)* Find the smallest index $\rho \in \{1, \dots, K\}$ such that:

$$u_\rho - \frac{1}{\rho} \left(\sum_{j=1}^{\rho} u_j - 1 \right) > 0. \quad (27)$$

- iii)* Compute the threshold:

$$\tau = \frac{1}{\rho} \left(\sum_{j=1}^{\rho} u_j - 1 \right). \quad (28)$$

- iv)* The projection is then:

$$\mathbf{x}^* = \max(\mathbf{z} - \tau, 0). \quad (29)$$

Graph inpainting. During generation, we use inpainting to enforce constraints: *i)* budget splits are fixed to 1 for unexpanded clusters or those with a budget of 1, *ii)* equal splits are enforced for expanded clusters with a budget of 2, *iii)* clusters of size one are not allowed to expand; and *iv)* features of non-expanded clusters are copied unchanged.

E.2 MODEL ARCHITECTURE

Our method represents the expansion numbers for left and right nodes, along with edge presence, as attributes of the bipartite graph. To model the distribution $p(\mathbf{v}_L^{(l)}, \mathbf{v}_R^{(l)}, \mathbf{e}^{(l)}, \mathbf{f}, \mathbf{F}_L^{\text{refine}} \mid \tilde{B}^{(l)})$, we adopt an endpoint-parameterized flow-matching framework Lipman et al. (2022). Within this framework, the attributes—namely, the expansion vectors and edge indicators—are corrupted with noise, and a denoising network is trained to reconstruct the original values.

The denoising network is structured as follows:

1. **Positional encoding:** Node positions within the graph are encoded using SignNet Lim et al. (2022). These encodings are replicated according to the respective expansion numbers.

- 1080 2. **Attribute embedding:** Five separate linear layers are used to embed the bipartite graph attributes:
 1081 left node features, right node features, edge features, node-specific features, and hyperedge-
 1082 specific features. FiLM conditioning Perez et al. (2018) is applied to incorporate contextual
 1083 information into node and hyperedge features. Node budgets are embedded using sinusoidal
 1084 positional encodings Vaswani et al. (2017).
- 1085 3. **Feature concatenation:**
- 1086 • For each left and right node, embeddings are concatenated with positional encodings and the
 1087 desired reduction fraction. Left nodes also receive the node budget embedding.
 - 1088 • If node features are present, they are appended to the left nodes. Likewise, hyperedge
 1089 features are appended to the right nodes when available.
 - 1090 • For edges, embeddings include edge features, concatenated positional encodings of the
 1091 incident nodes, and the reduction fraction.
- 1092 4. **Graph processing:** The attribute-enriched bipartite graph is processed through a stack of sparse
 1093 PPGN layers, following the architecture from Bergmeister et al. (2024).
- 1094 5. **Output prediction:** The final graph representations are passed through three linear projection
 1095 heads to generate outputs.
- 1096 • Left node head: Predicts expansion values, budget splits, and refined node features.
 - 1097 • Right node head: Predicts hyperedge expansions and refined hyperedge features.
 - 1098 • Edge head: Predicts edge existence.
- 1099

1100 E.3 ADDITIONAL DETAILS

1101 **Perturbed expansion.** Building on Bergmeister et al. (2024); Gailhard et al. (2025), we augment
 1102 Definitions 4 and 6—which are sufficient for reversing coarsening steps—with additional randomness
 1103 to enhance generative quality. This modification is especially beneficial in low-data regimes where
 1104 overfitting is a concern. Specifically, we introduce a probabilistic mechanism that supplements the set
 1105 of edges \mathcal{E} by randomly adding edges between node pairs on opposite sides of the bipartite graph that
 1106 are within a fixed distance in B . The following definition extends the expansion process (Definition
 1107 4) to include this stochastic component.

1108 **Definition 13** (Perturbed hypergraph expansion). Let $B = (\mathcal{V}_L, \mathcal{V}_R, \mathcal{E})$ be a bipartite graph, and
 1109 let $\mathbf{v}_L \in \mathbb{N}^{|\mathcal{V}_L|}$ and $\mathbf{v}_R \in \mathbb{N}^{|\mathcal{V}_R|}$ denote the left and right cluster size vectors. For a given radius
 1110 $r \in \mathbb{N}$ and probability $0 \leq p \leq 1$, we construct \tilde{B} as in Definition 4. Additionally, for each pair of
 1111 distinct nodes $\mathbf{v}_L(p) \in \mathcal{V}_L$ and $\mathbf{v}_R(q) \in \mathcal{V}_R$ that are within a distance of at most $2r + 1$ in B , we
 1112 independently add an edge $e_{\{p_i, q_j\}}$ to $\tilde{\mathcal{E}}$ with probability p .

1113 **Spectral conditioning.** In line with Martinkus et al. (2022); Bergmeister et al. (2024), we incorporate
 1114 spectral information—specifically, the principal eigenvalues and eigenvectors of the *normalized*
 1115 Laplacian—as a form of conditioning during the generative process. This technique has been shown
 1116 to improve the quality of generated graphs. To generate $B^{(l)}$ from its coarser form $B^{(l+1)}$, we
 1117 leverage the approximate spectral invariance under coarsening. We compute the k smallest non-
 1118 zero eigenvalues and their corresponding eigenvectors from the normalized Laplacian matrix $\mathcal{L}^{(l+1)}$
 1119 of $B^{(l+1)}$. These eigenvectors are processed using SignNet Lim et al. (2022) to produce node
 1120 embeddings for $B^{(l+1)}$. These embeddings are then propagated to the expanded nodes of $B^{(l)}$,
 1121 helping to preserve structural coherence and facilitate cluster identification. The hyperparameter k
 1122 controls the number of spectral components used.

1123 **Minibatch OT-coupling.** Minibatch OT-coupling Tong et al. (2024); Pooladian et al. (2023) acceler-
 1124 ates training by jointly sampling priors and targets and reindexing priors to minimize input–output
 1125 distance, leading to smoother flows and fewer inference steps. As detailed in Section 3.5, we adapt
 1126 this idea by treating each side of the bipartite graph as a minibatch and applying OT-coupling within
 1127 them. To preserve the distribution, we restrict permutations to groups of *equivalent nodes* (identical
 1128 topology and features, differing only in noise). For efficiency, we assume equivalence only among
 1129 nodes from the same cluster expansion. Since each cluster produces two or three children, coupling
 1130 reduces to evaluating two or six possible permutations per cluster, which can be efficiently parallelized
 1131 with tensors (Algorithm 2).

1134
 1135
 1136
 1137
 1138
 1139
 1140
 1141
 1142
 1143
 1144
 1145
 1146
 1147
 1148
 1149
 1150
 1151
 1152
 1153
 1154
 1155
 1156
 1157
 1158
 1159
 1160
 1161
 1162
 1163
 1164
 1165
 1166
 1167
 1168
 1169
 1170
 1171
 1172
 1173
 1174
 1175
 1176
 1177
 1178
 1179
 1180
 1181
 1182
 1183
 1184
 1185
 1186
 1187

Algorithm 2 Minibatch OT-coupling for coarsening/expansion strategy

Require: Expanded bipartite representation B with clusters $\{V^{(p)}\}$, each $V^{(p)}$ containing 1 or 2 nodes; target samples $\{x_i\}$

Ensure: Reindexed noise samples $\{\tilde{z}_i\}$

- 1: Sample noise $\{z_i\}$ for each node in B
- 2: **for all** clusters $V^{(p)} \in B$ **do**
- 3: **if** $|V^{(p)}| = 1$ **then**
- 4: No reassignment needed for singleton cluster
- 5: **else if** $|V^{(p)}| = 2$ **then**
- 6: Let i, j be the indices of the two nodes in $V^{(p)}$
- 7: Let x_i, x_j be their corresponding targets
- 8: Compute normal order cost:

$$C_{\text{normal}} \leftarrow \|z_i - x_i\|^2 + \|z_j - x_j\|^2$$
- 9: Compute swapped order cost:

$$C_{\text{swap}} \leftarrow \|z_j - x_i\|^2 + \|z_i - x_j\|^2$$
- 10: **if** $C_{\text{swap}} < C_{\text{normal}}$ **then**
- 11: Swap $z_i \leftrightarrow z_j$
- 12: **end if**
- 13: **end if**
- 14: **end for**
- 15: **return** $\{\tilde{z}_i\}$ (reassigned noise samples)

F TRAINING AND SAMPLING PROCEDURES

In this section, we present the complete training and inference procedures, detailed in Algorithms 4 and 5. Both pipelines rely on node embeddings produced by Algorithm 3.

Algorithm 3 Node embedding computation: Here we describe the way the left and right side node embeddings are computed for a given bipartite representation of a hypergraph. Embeddings are computed for the input bipartite representation and then replicated according to the cluster size vectors.

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 vector \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

```

1: function EMBEDDINGS( $B = (\mathcal{V}_L, \mathcal{V}_R, \mathcal{E}), \text{SignNet}_\theta, \mathbf{v}_L, \mathbf{v}_R$ )
2:   if  $k = 0$  then
3:      $\mathbf{H} = [h^{(1)}, \dots, h^{(|\mathcal{V}|)}] \stackrel{i.i.d.}{\sim} \mathcal{N}(0, I)$  ▷ Sample random embeddings
4:   else
5:     if  $k < |\mathcal{V}|$  then
6:        $[\lambda_1, \dots, \lambda_k], [u_1, \dots, u_k] \leftarrow \text{EIG}(B)$  ▷ Compute  $k$  spectral features
7:     else
8:        $[\lambda_1, \dots, \lambda_{|\mathcal{V}_L|+|\mathcal{V}_R|-1}], [u_1, \dots, u_{|\mathcal{V}_L|+|\mathcal{V}_R|-1}] \leftarrow \text{EIG}(B)$  ▷ Compute
9:        $|\mathcal{V}_L| + |\mathcal{V}_R| - 1$  spectral features
10:       $[\lambda_{|\mathcal{V}_L|+|\mathcal{V}_R|}, \dots, \lambda_k], [u_{|\mathcal{V}_L|+|\mathcal{V}_R|}, \dots, u_k] \leftarrow [0, \dots, 0], [0, \dots, 0]$  ▷ Pad with zeros
11:    end if
12:     $\mathbf{H} = [h^{(1)}, \dots, h^{(|\mathcal{V}_L|+|\mathcal{V}_R|)}] \leftarrow \text{SignNet}_\theta([\lambda_1, \dots, \lambda_k], [u_1, \dots, u_k], B)$ 
13:     $\tilde{B} = (\mathcal{V}_L^{(1)} \cup \dots \cup \mathcal{V}_L^{(p_L)}, \mathcal{V}_R^{(1)} \cup \dots \cup \mathcal{V}_R^{(p_R)}, \tilde{\mathcal{E}}) \leftarrow \tilde{B}(B, \mathbf{v}_L, \mathbf{v}_R)$  ▷ Expand as per
14:    Definition 4
15:    set  $\tilde{B}$  s.t. for all  $p_L \in [|\mathcal{V}_L|]$  and all  $p_R \in [|\mathcal{V}_R|]$ : for all  $\mathbf{v}_L^{(p_L)} \in \mathcal{V}_L^{(p_L)}, \tilde{\mathbf{H}}[p_L] = \mathbf{H}[p_L]$  and
16:    for all  $\mathbf{v}_R^{(p_R)} \in \mathcal{V}_R^{(p_R)}, \tilde{\mathbf{H}}[p_R] = \mathbf{H}[p_R]$  ▷ Replicate embeddings
17:    return  $\tilde{\mathbf{H}}$ 
18: end function

```

1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295

Algorithm 4 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 $\mathcal{D} = \{H_1, \dots, H_N\}$, denoising model GNN_θ , spectral feature model SignNet_θ
Output: trained model parameters θ

```

1: function TRAIN( $\mathcal{D}$ ,  $\text{GNN}_\theta$ ,  $\text{SignNet}_\theta$ )
2:   while not converged do
3:      $H \sim \text{Uniform}(\mathcal{D})$  ▷ Sample graph
4:      $(B^{(0)}, \dots, B^{(L)}) \leftarrow \text{RndRedSeq}(H)$  ▷ Sample coarsening sequence by Algorithm 1
5:      $l \sim \text{Uniform}(\{0, \dots, L\})$  ▷ Sample level
6:     if  $l = 0$  then
7:        $\mathbf{v}_L^{(0)} \leftarrow 1, \mathbf{v}_R^{(0)} \leftarrow 1$ 
8:     else
9:       set  $\mathbf{v}_L^{(l)}$  and  $\mathbf{v}_R^{(l)}$  such that the node sets of  $\tilde{B}(B^{(l)}, \mathbf{v}_L^{(l)}, \mathbf{v}_R^{(l)})$  equals that of  $B^{(l-1)}$ 
10:    end if
11:    if  $l = L$  then
12:       $B^{(l+1)} \leftarrow B^{(l)} = (\{1\}, \{2\}, \{(1, 2)\}, \mathbf{b} = \text{size}(H), \mathbf{F}_L = 0, \mathbf{F}_R = 0)$ 
13:       $\mathbf{v}_L^{(l+1)} \leftarrow 1$ 
14:       $\mathbf{v}_R^{(l+1)} \leftarrow 1$ 
15:       $\mathbf{e}^{(l)} \leftarrow 1$ 
16:    end if
17:    set  $\mathbf{e}^{(l)}, \mathbf{f}, \mathbf{F}_L^{\text{refine}}$  and  $\mathbf{F}_R^{\text{refine}}$  such that  $B(\tilde{B}(B^{(l+1)}, \mathbf{v}_L^{(l+1)}, \mathbf{v}_R^{(l+1)}), \mathbf{e}^{(l)}, \mathbf{f}, \mathbf{F}_L^{\text{refine}}, \mathbf{F}_R^{\text{refine}}) =$ 
18:       $B^{(L)}$ 
19:       $\mathbf{H}^{(l)} \leftarrow \text{Embeddings}(B^{(l+1)}, \text{SignNet}_\theta, \mathbf{v}_L^{(l+1)}, \mathbf{v}_R^{(l+1)})$  ▷ Compute node embeddings
20:       $\hat{\rho} \leftarrow 1 - (n^{(l)}/n^{(l-1)})$ , with  $n^{(l)}$  and  $n^{(l-1)}$  being the size of the left side of  $B^{(l)}$  and
21:       $B^{(l-1)}$ 
22:       $D_\theta \leftarrow \text{GNN}_\theta(\cdot, \cdot, \tilde{B}^{(l)}, \mathbf{H}^{(l)}, n^{(0)}, \rho)$ , where  $n^{(0)}$  is the size of the left side of  $B^{(0)}$ 
23:      take gradient descent step on  $\nabla_\theta \text{DiffusionLoss}(\mathbf{v}_L^{(L)}, \mathbf{v}_R^{(L)}, \mathbf{e}^{(l)}, \mathbf{f}, \mathbf{F}_L^{\text{refine}}, \mathbf{F}_R^{\text{refine}}, D_\theta)$ 
24:    end while
25:  return  $\theta$ 
26: end function

```

1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349

Algorithm 5 End-to-end sampling procedure with deterministic expansion size: This describes the sampling procedure. Note that this assumes that the maximum cluster sizes are 2 and 3, which is the case when using edges of the clique representation as the contraction set family for model training.

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 $|\mathcal{V}| = N$

- 1: **function** `SAMPLE($N, \text{GNN}_\theta, \text{SignNet}_\theta$)`
- 2: $B = (\mathcal{V}_L, \mathcal{V}_R, \mathcal{E}, \mathbf{f}, \mathbf{F}_L^{\text{refine}}, \mathbf{F}_R^{\text{refine}}) \leftarrow (\{1\}, \{2\}, \{(1, 2)\}, N, 0, 0)$ \triangleright Start with a minimal bipartite graph
- 3: $\mathbf{v}_L \leftarrow [1], \mathbf{v}_R \leftarrow [1]$ \triangleright Initial cluster size vectors
- 4: **while** $|\mathcal{V}_L| < N$ **do**
- 5: $\mathbf{H} \leftarrow \text{Embeddings}(B, \text{SignNet}_\theta, \mathbf{v}_L, \mathbf{v}_R)$ \triangleright Compute node embeddings
- 6: $n \leftarrow \|\mathbf{v}_L\|_1$
- 7: $\rho \sim \text{Uniform}([\rho_{\min}, \rho_{\max}])$ \triangleright random reduction fraction
- 8: set n^+ s.t. $n^+ = \lceil \rho(n + n^+) \rceil$ \triangleright number of left side nodes to add
- 9: $n^+ \leftarrow \min(n^+, N - n)$ \triangleright ensure not to exceed target size
- 10: $\hat{\rho} \leftarrow 1 - (n / (n + n^+))$ \triangleright actual reduction fraction
- 11: $D_\theta \leftarrow \text{GNN}_\theta(\cdot, \cdot, \tilde{B}(B, \mathbf{v}_L, \mathbf{v}_R), \mathbf{H}, N, \hat{\rho})$
- 12: $(\mathbf{v}_L)_0, (\mathbf{v}_R)_0, (\mathbf{e})_0, \mathbf{f}, \mathbf{F}_L^{\text{refine}}, \mathbf{F}_R^{\text{refine}} \leftarrow \text{Sample}(D_\theta)$ \triangleright Sample features
- 13: set \mathbf{v}_L s.t. for $i \in [n]$: $\mathbf{v}_L[i] = 2$ if $|\{j \in [n] \mid (\mathbf{v}_L)_0[j] \geq (\mathbf{v}_L)_0[i]\}| \geq n^+$ and $v[i] = 1$
- 14: **otherwise** set \mathbf{v}_R s.t. for $i \in [(\mathbf{v}_R)_0]$: $\mathbf{v}_R[i] = 1$ if $(\mathbf{v}_R)_0 < 1.66$, $\mathbf{v}_R[i] = 2$ if $(\mathbf{v}_R)_0 < 2.33$ and $\mathbf{v}_R[i] = 3$ otherwise
- 15: set \mathbf{e} s.t. for $i \in [(\mathbf{e})_0]$: $\mathbf{e}[i] = 1$ if $(\mathbf{e})_0 > 0.5$ and $\mathbf{e}[i] = 0$ otherwise
- 16: $B = (\mathcal{V}_L, \mathcal{V}_R, \mathcal{E}) \leftarrow B(\tilde{B}, \mathbf{e}, \mathbf{f}, \mathbf{F}_L^{\text{refine}}, \mathbf{F}_R^{\text{refine}})$ \triangleright Refine as per Definition 6
- 17: **end while**
- 18: build H from its bipartite representation B
- 19: **return** H
- 20: **end function**

G COMPLEXITY ANALYSIS

In this section, we investigate the asymptotic complexity of our proposed algorithm, which extends the methodology introduced by Bergmeister et al. (2024) and Gailhard et al. (2025). To construct a hypergraph comprising n nodes, m hyperedges, and k incidences, the algorithm sequentially produces a series of bipartite graphs $B^{(L)} = (\{1\}, \{2\}, \{(1, 2)\}), B^{(L-1)}, \dots, B^{(0)} = B$, where the final graph B corresponds to the bipartite representation of the generated hypergraph. We use n , m , and k to denote, respectively, the number of nodes, hyperedges, and incidences in the hypergraph, and as the number of left-side nodes, right-side nodes, and edges in the corresponding bipartite graph.

For each level $0 \leq l < L$ of the sequence, the number of left-side nodes in $B^{(l)}$, denoted n_l , satisfies $n_l \geq (1 + \epsilon)n_{l-1}$ for some $\epsilon > 0$ (e.g., $\epsilon = \text{reduction_frac}/(1 - \text{reduction_frac})$). This implies an upper bound on the number of steps in the expansion sequence: $\lceil \log_{1+\epsilon} n \rceil \in \mathcal{O}(\log n)$. Since the expansion process only increases node counts, all B_l graphs contain fewer than n left-side and m right-side nodes. The number of edges, however, may temporarily exceed k , as the intermediate bipartite graphs may include additional edges removed in later refinements. Still, because the coarsening during training consistently reduces incidences, the model is expected to learn accurate edge refinement and avoid such accumulation. Consequently, we assume $k_l \leq k$ and $m_l \leq m$ for all $0 \leq l \leq L$.

Next, we assess the computational cost of generating a single expansion step. At level $l = L$, this consists of creating a pair of connected nodes, initializing features as matrices of zeros, initializing budget as the targeted node count, and predicting the expansion vectors \mathbf{v}_L and \mathbf{v}_R —a process with constant complexity $\mathcal{O}(1)$. For levels $0 \leq l < L$, given $B^{(l+1)}$ and expansion vectors $\mathbf{v}_L^{(l+1)}$, $\mathbf{v}_R^{(l+1)}$, the algorithm constructs the expanded bipartite graph $\tilde{B}(B^{(l+1)}, \mathbf{v}_L^{(l+1)}, \mathbf{v}_R^{(l+1)})$ in $\mathcal{O}(n + m)$ time. It then samples $\mathbf{v}_L^{(l)}$, $\mathbf{v}_R^{(l)}$, $\mathbf{e}^{(l)}$, \mathbf{f} , $\mathbf{F}_L^{\text{refine}}$, and $\mathbf{F}_R^{\text{refine}}$, and constructs the refined graph $B^{(l)} = B(\tilde{B}^{(l)}, \mathbf{e}^{(l)}, \mathbf{f}, \mathbf{F}_L^{\text{refine}}, \mathbf{F}_R^{\text{refine}})$. Letting v_{\max}^L and v_{\max}^R be the maximum cluster sizes, the incidence count in $\tilde{B}^{(l)}$ is bounded by $k_l \leq k_{l+1} v_{\max}^L v_{\max}^R$.

The sampling process queries a denoising model a constant number of times per step. The complexity is thus governed by the architecture. In our case, since bipartite graphs are triangle-free, the *Local PPGN* model Bergmeister et al. (2024) has linear complexity $\mathcal{O}(n + m + k)$. Embedding computation for $B^{(l)}$ similarly costs $\mathcal{O}(n + m + k)$. This includes calculating the top K eigenvalues/eigenvectors of the Laplacian via the method from Vishnoi (2013), with complexity $\mathcal{O}(K(n_{l+1} + m_{l+1} + k_{l+1}))$, and embedding via *SignNet*, also linear in graph size due to fixed K .

The final transformation from the bipartite graph to a hypergraph—by collapsing right-side nodes into hyperedges—has a cost of $\mathcal{O}(m + k)$. Under these assumptions, the total complexity to generate a hypergraph H with n nodes, m hyperedges, and k incidences is $\mathcal{O}(n + m + k)$.

H DETAILED NUMERICAL RESULTS

In this section, we present detailed numerical results for all datasets and metrics described in Appendix D. Reported values of the form $a \pm b$ indicate the mean a and twice the standard deviation b computed over 5 runs. The best and second-best results are highlighted in **bold** and underlined, respectively.

H.1 COMPARISONS WITH THE BASELINES

SBM Hypergraphs								
Method	Valid \uparrow	NumDiff \downarrow	Deg \downarrow	EdgeSize \downarrow	Spectral \downarrow	Harmonic \downarrow	Closeness \downarrow	Betweenness \downarrow
HyperPA Do et al. (2020)	2.5	<u>0.075</u>	4.062	0.407	0.273	77.840	0.074	0.008
VAE Kingma & Welling (2013)	0.0	0.375	1.280	1.059	0.024	6.543	0.007	0.006
GAN Goodfellow et al. (2020)	0.0	1.200	2.106	1.203	0.059	10.700	0.076	0.012
Diffusion Ho et al. (2020)	0.0	0.150	1.717	1.390	0.031	13.940	0.040	0.004
HYGENE Gailhard et al. (2025)	<u>65.0</u>	<u>0.525</u>	0.321	0.002	<u>0.010</u>	2.990	0.016	0.000
FAHNES	87.8\pm3.1	0.029\pm0.009	<u>0.846\pm0.457</u>	<u>0.005\pm0.003</u>	0.006\pm0.004	<u>6.410\pm3.124</u>	<u>0.009\pm0.006</u>	<u>0.003\pm0.001</u>
Ego Hypergraphs								
Method	Valid \uparrow	NumDiff \downarrow	Deg \downarrow	EdgeSize \downarrow	Spectral \downarrow	Harmonic \downarrow	Closeness \downarrow	Betweenness \downarrow
HyperPA Do et al. (2020)	0.0	35.830	2.590	0.423	0.237	143.000	0.354	0.002
VAE Kingma & Welling (2013)	0.0	47.580	0.803	1.458	0.133	38.950	0.558	0.019
GAN Goodfellow et al. (2020)	0.0	60.350	0.917	1.665	0.230	41.800	0.612	0.015
Diffusion Ho et al. (2020)	0.0	<u>4.475</u>	3.984	2.985	0.190	6.911	0.407	0.009
HYGENE Gailhard et al. (2025)	<u>90.0</u>	<u>12.550</u>	0.063	<u>0.220</u>	0.004	<u>5.790</u>	<u>0.025</u>	0.000
FAHNES	99.5\pm1.1	0.128\pm0.171	<u>0.124\pm0.086</u>	<u>0.155\pm0.067</u>	0.004\pm0.003	2.703\pm3.468	0.003\pm0.005	0.000\pm0.000
Tree Hypergraphs								
Method	Valid \uparrow	NumDiff \downarrow	Deg \downarrow	EdgeSize \downarrow	Spectral \downarrow	Harmonic \downarrow	Closeness \downarrow	Betweenness \downarrow
HyperPA Do et al. (2020)	0.0	2.350	0.315	0.284	0.159	5.941	0.477	0.168
VAE Kingma & Welling (2013)	0.0	9.700	0.072	0.480	0.124	3.869	0.280	0.139
GAN Goodfellow et al. (2020)	0.0	6.000	0.151	0.469	0.089	2.198	0.201	0.124
Diffusion Ho et al. (2020)	0.0	2.225	1.718	1.922	0.127	8.565	0.353	0.139
HYGENE Gailhard et al. (2025)	<u>77.5</u>	0.000	<u>0.059</u>	<u>0.108</u>	<u>0.012</u>	<u>1.099</u>	<u>0.041</u>	<u>0.016</u>
FAHNES	89.7\pm6.0	0.000\pm0.000	<u>0.022\pm0.022</u>	<u>0.030\pm0.034</u>	0.003\pm0.002	<u>0.171\pm0.106</u>	<u>0.014\pm0.006</u>	<u>0.014\pm0.004</u>
ModelNet Bookshelf								
Method	NumDiff \downarrow	Deg \downarrow	EdgeSize \downarrow	Spectral \downarrow	Harmonic \downarrow	Closeness \downarrow	Betweenness \downarrow	
HyperPA Do et al. (2020)	8.025	7.562	0.044	<u>0.048</u>	877.500	0.211	0.005	
VAE Kingma & Welling (2013)	47.450	6.190	1.520	0.190	113.600	<u>0.145</u>	<u>0.003</u>	
GAN Goodfellow et al. (2020)	0.000	397.200	46.300	0.476	670.100	0.707	0.007	
Diffusion Ho et al. (2020)	0.000	20.360	2.346	0.079	264.100	0.239	0.006	
HYGENE Gailhard et al. (2025)	69.730	1.050	<u>0.034</u>	0.068	27.400	0.204	0.004	
FAHNES	<u>0.135\pm0.276</u>	<u>2.980\pm2.107</u>	0.020\pm0.025	0.024\pm0.015	<u>46.614\pm58.803</u>	0.086\pm0.030	0.001\pm0.001	
ModelNet Piano								
Method	NumDiff \downarrow	Deg \downarrow	EdgeSize \downarrow	Spectral \downarrow	Harmonic \downarrow	Closeness \downarrow	Betweenness \downarrow	
HyperPA Do et al. (2020)	0.825	9.254	0.023	<u>0.067</u>	77.840	<u>0.236</u>	0.004	
VAE Kingma & Welling (2013)	75.350	8.060	1.686	0.396	184.300	0.241	0.003	
GAN Goodfellow et al. (2020)	0.000	409.000	86.380	0.697	622.200	0.738	0.005	
Diffusion Ho et al. (2020)	<u>0.050</u>	20.900	4.192	0.113	289.300	0.303	0.004	
HYGENE Gailhard et al. (2025)	42.520	<u>6.290</u>	<u>0.027</u>	0.117	155.000	0.285	0.002	
FAHNES	<u>0.846\pm1.009</u>	3.265\pm1.954	<u>0.042\pm0.056</u>	0.040\pm0.026	<u>119.158\pm81.023</u>	0.123\pm0.119	0.002\pm0.002	

Table 6: Detailed numerical results for unfeatured hypergraphs.

ManifoldNet Airplane								
Method	ChamDist	NumDiff \downarrow	Deg \downarrow	EdgeSize \downarrow	Spectral \downarrow	Harmonic \downarrow	Closeness \downarrow	Betweenness \downarrow
Sequential	0.143	0.367	0.801	0.004	0.007	4.964	0.087	0.006
FAHNES	0.048\pm0.003	0.017\pm0.070	0.218\pm0.085	0.004\pm0.007	<u>0.010\pm0.003</u>	2.428\pm1.455	0.032\pm0.007	0.003\pm0.001
ManifoldNet Bench								
Method	ChamDist	NumDiff \downarrow	Deg \downarrow	EdgeSize \downarrow	Spectral \downarrow	Harmonic \downarrow	Closeness \downarrow	Betweenness \downarrow
Sequential	0.117	0.078	0.332	0.011	0.015	3.131	0.035	0.007
FAHNES	0.064\pm0.005	0.060\pm0.149	<u>0.349\pm0.454</u>	0.009\pm0.017	<u>0.017\pm0.011</u>	<u>5.069\pm8.171</u>	<u>0.046\pm0.040</u>	0.004\pm0.003

Table 7: Detailed numerical results for 3D meshes.

Airplane Point Clouds								
Method	ChamDist ↓	NumDiff ↓	Wavelet ↓	Orbit ↓	Clustering ↓	Deg ↓	Spectral ↓	Ratio ↓
DiGress Vignac et al. (2023)	OOM							
DeFoG Qin et al. (2024)	OOM							
FAHNES	0.094±0.006	0.000±0.000	0.004±0.001	0.074±0.104	0.264±0.254	0.002±0.002	0.005±0.004	67.311±44.903

Bench Point Clouds								
Method	ChamDist ↓	NumDiff ↓	Wavelet ↓	Orbit ↓	Clustering ↓	Deg ↓	Spectral ↓	Ratio ↓
DiGress Vignac et al. (2023)	OOM							
DeFoG Qin et al. (2024)	OOM							
FAHNES	0.130±0.001	0.000±0.000	0.003±0.000	0.013±0.004	0.229±0.071	0.000±0.000	0.004±0.000	73.279±22.287

Table 8: Detailed numerical results for graph point cloud datasets.

H.2 ABLATION STUDIES

SBM Hypergraphs									
Node Budget	Minibatch OT	Valid ↑	NumDiff ↓	Deg ↓	EdgeSize ↓	Spectral ↓	Harmonic ↓	Closeness ↓	Betweenness ↓
✓	✓	87.8±3.1	0.029±0.009	0.846±0.457	0.005±0.003	0.006±0.004	6.410±3.124	0.009±0.006	0.003±0.001
✗	✓	85.3±5.9	0.044±0.078	0.856±0.636	0.023±0.022	0.006±0.004	6.210±4.649	0.009±0.013	0.003±0.003
✓	✗	86.7±6.7	<u>0.039±0.014</u>	0.910±0.363	<u>0.006±0.004</u>	0.006±0.004	6.815±1.913	0.010±0.005	0.004±0.001
✗	✗	84.6±4.6	0.049±0.045	0.916±0.749	0.047±0.070	0.007±0.007	6.665±5.199	0.012±0.014	0.004±0.004

Ego Hypergraphs									
Node Budget	Minibatch OT	Valid ↑	NumDiff ↓	Deg ↓	EdgeSize ↓	Spectral ↓	Harmonic ↓	Closeness ↓	Betweenness ↓
✓	✓	99.5±1.1	0.128±0.171	<u>0.124±0.086</u>	<u>0.155±0.067</u>	0.004±0.003	2.703±3.468	0.003±0.005	0.000
✗	✓	100.0±0.0	0.118±0.158	0.140±0.104	0.133±0.106	0.005±0.003	3.426±2.629	0.000	0.000
✓	✗	<u>99.9±0.4</u>	0.073±0.050	0.237±0.465	0.193±0.175	0.007±0.012	4.521±5.629	0.000±0.002	0.000
✗	✗	99.5±1.1	<u>0.117±0.155</u>	0.110±0.111	0.189±0.084	0.004±0.003	<u>2.765±1.684</u>	0.001±0.004	0.000

Tree Hypergraphs									
Node Budget	Minibatch OT	Valid ↑	NumDiff ↓	Deg ↓	EdgeSize ↓	Spectral ↓	Harmonic ↓	Closeness ↓	Betweenness ↓
✓	✓	89.7±6.0	0.000	<u>0.022±0.022</u>	0.030±0.034	0.003±0.002	0.171±0.106	<u>0.014±0.006</u>	0.014±0.004
✗	✓	<u>90.7±6.7</u>	0.000	0.024±0.019	0.067±0.016	<u>0.004±0.001</u>	0.315±0.098	0.018±0.012	0.014±0.008
✓	✗	89.3±3.6	0.000	0.019±0.013	<u>0.047±0.077</u>	0.005±0.011	0.169±0.106	0.041±0.082	<u>0.013±0.010</u>
✗	✗	95.6±3.7	0.000	0.042±0.042	0.077±0.091	0.005±0.003	0.217±0.169	0.013±0.013	0.008±0.011

ModelNet Bookshelf									
Node Budget	Minibatch OT	NumDiff ↓	Deg ↓	EdgeSize ↓	Spectral ↓	Harmonic ↓	Closeness ↓	Betweenness ↓	
✓	✓	0.135±0.276	<u>2.980±2.107</u>	0.020±0.025	<u>0.024±0.015</u>	46.614±58.803	0.086±0.030	0.001±0.001	
✗	✓	0.940±0.917	3.040±1.446	0.089±0.099	0.032±0.013	59.300±108.101	0.137±0.042	0.003±0.001	
✓	✗	<u>0.265±0.496</u>	4.400±1.635	<u>0.025±0.021</u>	0.014±0.007	<u>54.558±30.276</u>	<u>0.102±0.020</u>	0.001±0.001	
✗	✗	1.325±1.631	2.820±0.958	0.055±0.077	0.031±0.009	79.889±98.721	0.135±0.012	0.003±0.001	

ModelNet Piano									
Node Budget	Minibatch OT	NumDiff ↓	Deg ↓	EdgeSize ↓	Spectral ↓	Harmonic ↓	Closeness ↓	Betweenness ↓	
✓	✓	0.846±1.009	3.265±1.954	0.042±0.056	0.040±0.026	<u>119.158±81.023</u>	0.123±0.119	0.002±0.002	
✗	✓	3.622±1.822	<u>3.358±1.772</u>	<u>0.054±0.080</u>	0.055±0.040	133.806±137.603	0.155±0.058	0.012±0.033	
✓	✗	<u>3.155±3.637</u>	5.250±1.087	0.066±0.168	0.030±0.048	154.211±383.979	0.188±0.164	0.002±0.002	
✗	✗	5.490±8.847	3.733±2.989	0.107±0.215	<u>0.036±0.016</u>	97.724±198.774	<u>0.138±0.129</u>	0.002±0.002	

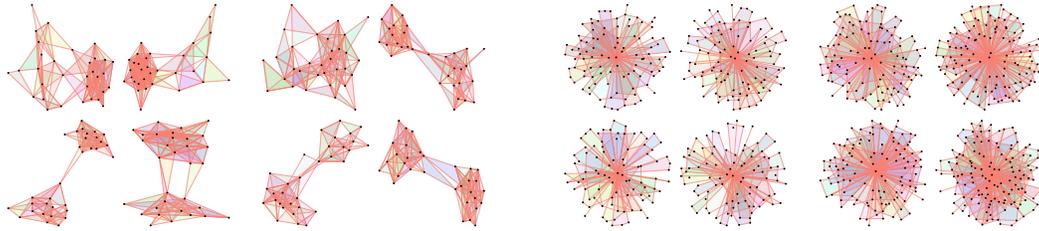
Table 9: Detailed numerical results for ablation studies on unfeatured hypergraphs.

ManifoldNet Bench									
Node Budget	Minibatch OT	ChamDist ↓	NumDiff ↓	Deg ↓	EdgeSize ↓	Spectral ↓	Harmonic ↓	Closeness ↓	Betweenness ↓
✓	✓	0.064±0.005	0.060±0.149	<u>0.349±0.454</u>	<u>0.009±0.017</u>	<u>0.017±0.011</u>	5.069±8.171	0.046±0.040	0.004±0.003
✗	✓	0.090±0.003	0.240±0.265	1.089±0.346	0.013±0.015	0.018±0.009	6.507±2.771	0.014±0.005	<u>0.005±0.002</u>
✓	✗	<u>0.085±0.056</u>	0.020±0.032	0.221±0.160	0.006±0.004	0.013±0.003	1.864±2.828	0.028±0.019	<u>0.005±0.006</u>
✗	✗	0.098±0.024	0.267±0.319	1.242±0.582	0.013±0.011	0.022±0.014	7.619±4.165	<u>0.015±0.016</u>	0.006±0.004

ManifoldNet Airplane									
Node Budget	Minibatch OT	ChamDist ↓	NumDiff ↓	Deg ↓	EdgeSize ↓	Spectral ↓	Harmonic ↓	Closeness ↓	Betweenness ↓
✓	✓	0.048±0.003	0.017±0.070	0.218±0.085	0.004±0.007	0.010±0.003	<u>2.428±1.455</u>	0.032±0.007	0.003±0.001
✗	✓	0.079±0.019	0.426±0.820	0.588±0.451	0.024±0.021	0.014±0.007	2.429±1.394	0.033±0.028	0.003±0.002
✓	✗	<u>0.050±0.005</u>	<u>0.052±0.065</u>	<u>0.235±0.063</u>	<u>0.014±0.027</u>	<u>0.012±0.005</u>	1.604±1.400	0.022±0.007	0.003±0.001
✗	✗	0.100±0.023	0.304±0.437	0.864±0.358	0.020±0.016	0.019±0.009	4.184±2.176	<u>0.025±0.009</u>	0.005±0.002

Table 10: Detailed numerical results for ablation studies on 3D meshes.

I COMPARISON BETWEEN TRAINING AND GENERATED SAMPLES



Train samples

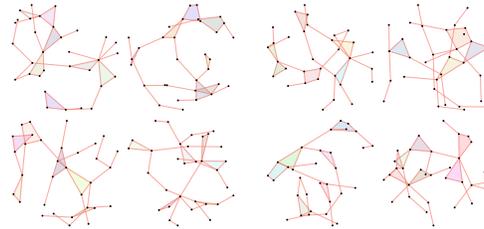
Generated samples

Train samples

Generated samples

(i) Stochastic Block Model hypergraphs.

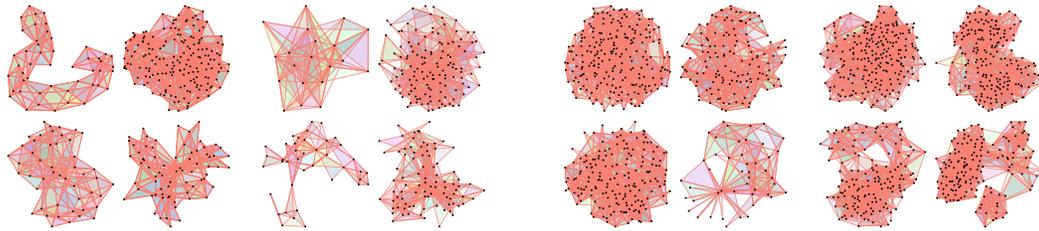
(ii) Ego hypergraphs.



Train samples

Generated samples

(iii) Tree hypergraphs.



Train samples

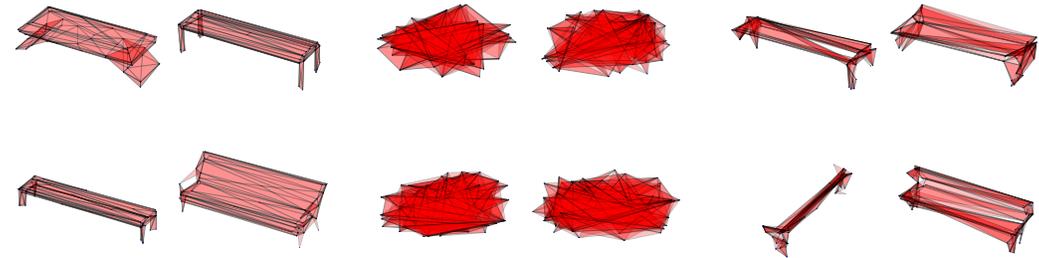
Generated samples

Train samples

Generated samples

(iv) Bookshelf meshes topology.

(v) Piano meshes topology



Train samples

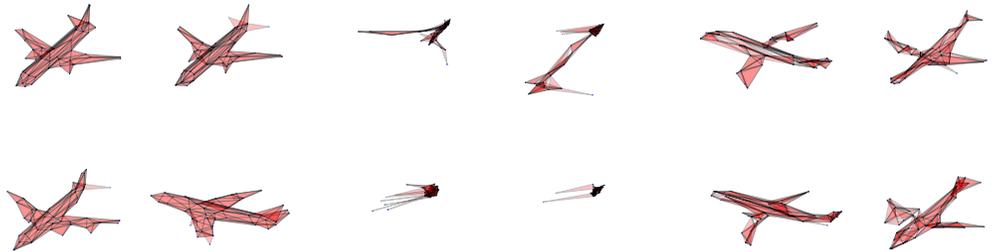
Generated samples (sequential)

Generated samples (joint)

(vi) Bench 3D meshes.

1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565

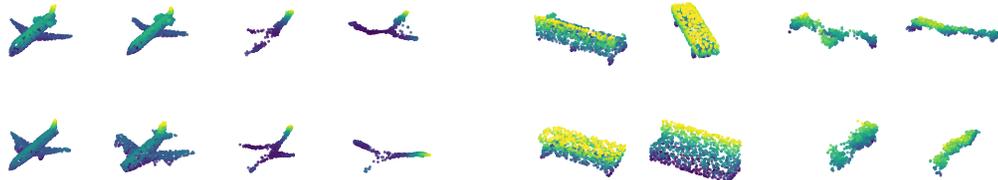
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619



Train samples

Generated samples (sequential)
(vii) Airplane 3D meshes.

Generated samples (joint)



Train samples Generated samples
(iii) ManifoldNet Airplane point clouds.

Train samples Generated samples
(iii) ManifoldNet Bench point clouds.