Hypergraph Generation with Latent Diffusion

Valerio Di Pasquale[1,†] **Alessia Antelmi**[2] **Mirko Polato**[2] **Carmine Spagnuolo**[1,†]

[1]Università degli Studi di Salerno, Fisciano, Italy
[2]Università degli Studi di Torino, Turin, Italy

[1]{vadipasquale, cspagnuolo}@unisa.it
[2]{alessia.antelmi, mirko.polato}@unito.it

hypernetwork-research-group/janus
hypernetwork-research-group.github.io/janus

Abstract

Hypergraphs capture high-order interactions that ordinary graphs cannot represent, yet generating realistic ones remains challenging because existing methods often rely on fixed structural assumptions or struggle to model coupled dependencies between nodes and hyperedges. In this work, we propose JANUS, a latent diffusion framework that decomposes an observed hypergraph into sub-hypergraphs, learns coupled node and hyperedge latent representations with a dual-view β-VAE, and generates new structures through paired latent diffusion with cross-view conditioning. The method supports unconstrained and node-set-constrained generation. We also introduce an evaluation protocol covering micro-, meso-, and macro-scale interactions, structural reconstruction, and high-order similarity measures. Across five real-world datasets and nine baselines, JANUS achieves the most stable performance, with strong gains in meso-scale and high-order fidelity; in the constrained setting, it consistently obtains the best reconstruction results.

1 Introduction

Hypergraphs provide a more accurate way to model high-order interactions that pairwise edges cannot represent [2]. These interactions occur in many real-world systems, including group communication, co-participation, co-purchasing, and biomedical associations [26]. Generating realistic hypergraphs is therefore essential for studying systems where high-order interactions are only partially observed, difficult to collect, or sensitive to share. The task of hypergraph generation aims to learn such high-order patterns from observed data and generate new hypergraphs that preserve their structural properties, enabling applications such as simulation, hypothesis testing, and synthetic data generation [15].

Most existing hypergraph generators rely on algorithmic or rule-based constructions to reproduce specific structural statistics [15, 3, 31]. These methods are often interpretable, but they typically rely on strong assumptions about the generation process. As a consequence, they may fail to capture the complex, multi-scale dependencies observed in real-world hypergraphs. Deep generative models offer a more flexible alternative [9]; however, their use for hypergraph generation remains largely unexplored [16].

Adapting deep generative models to hypergraph generation is challenging because hypergraphs contain two distinct types of entities, nodes and hyperedges, whose interaction increases the complexity of the system. Moreover, the number of possible hyperedges grows combinatorially with the number of nodes, making direct modeling in discrete space computationally intractable [8].

In this setting, diffusion-based generative models in continuous spaces provide a promising alternative, as they can learn smooth latent representations and improve generation accuracy when modeling

†Corresponding authors

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

complex structures [14, 22, 10]. In particular, latent diffusion models perform the diffusion process in a lower-dimensional latent space rather than in the original data space [41], making generation more efficient and scalable while preserving the expressive power needed to produce high-quality samples [50]. This approach is especially suitable for hypergraphs, as their discrete incidence structure can be embedded into a continuous latent space, thereby avoiding the need to apply diffusion directly to discrete structures.

Motivated by these advantages, in this work we address the problem of generating a hypergraph from observed data through a local-to-global approach. Rather than modeling the entire hypergraph at once, we first decompose the observed structure into sub-hypergraphs that capture local higher-order patterns. These local samples are then used to train a latent diffusion model that generates new sub-hypergraphs in the learned continuous latent space. Finally, we merge the generated sub-hypergraphs to construct new hypergraphs that preserve both local higher-order patterns and global structural properties, in line with prior approaches [34, 47]. Similar subsampling strategies have also been employed in image generation where the data space consists of a single instance [29].

Specifically, we propose JANUS, a dual-view latent diffusion framework for hypergraph generation that jointly generates nodes and hyperedge entities.[1] We evaluate JANUS in two settings: an *unconstrained* setting, where the model generates both nodes and hyperedges from scratch, and a *node-set-constrained setting*, where the model receives the node set as input. We assess performance using a novel evaluation framework that measures generation fidelity at multiple structural levels, including macro-, meso-, and micro-scale interactions, as well as the similarity between generated and real hypergraphs. Empirically, JANUS achieves the best performance among the evaluated baselines in the respective experimental settings.

In summary, our **contributions** include: (i) a dual-view latent diffusion framework that models nodes and hyperedges as coupled processes; (ii) a comprehensive multi-scale evaluation framework for hypergraph generation; and (iii) extensive experiments against nine baselines, demonstrating the superiority of our approach. Our code and datasets are publicly available in an open repository to support reproducibility: `https://github.com/hypernetwork-research-group/janus`.

2 Related work

Existing hypergraph generative models commonly fall into two categories: those that generate full hypergraphs versus sub-hypergraphs, and those that model static versus dynamic structures [31, 51]. Early methods mainly rely on random or algorithmic constructions that preserve selected structural statistics. For example, HYPERCL acts as a null model that preserves node degrees and hyperedge sizes [32]. Preferential attachment-based models extend this idea by capturing heavy-tailed patterns [15]: SUBSET SAMPLING reuses parts of existing hyperedges, thereby weakly preserving group structure; NAIVEPA applies preferential attachment at the individual node level and ignores group structure; and HYPERPA explicitly models group-wise attachment. Dynamic extensions include HYPERFF, which uses a forest-fire growth mechanism [28], and HMPA, which further incorporates community structure [17].

Another line of work focuses on reproducing specific higher-order properties. THERA targets the hypergraph transitivity [25]; HYPERLAP and HYPERLAP+ model hyperedge overlapness [32]; and HYPERDK reproduces local statistics at different orders to preserve specific structural patterns [37]. These approaches are often interpretable, as they directly link to predefined structural observables. However, their flexibility remains limited by the assumptions embedded in the generation process.

More recent learning-based methods aim to overcome these limitations by learning richer generative patterns from data. DGMH introduces a deep generative model based on hyperedge embeddings [52], while HYGENE applies a diffusion process to the bipartite representation of the hypergraph [16]. HYREC adopts a Kronecker-based formulation to support scalable generation [11].

Generation with a constrained node set. Fewer methods address the setting in which the node set is fixed and only the hyperedge structure is generated. HYPERPLR follows a projection-learning-reconstruction paradigm: it projects the hypergraph onto a weighted graph, learns latent representations, and reconstructs hyperedges through a clique-cover-style procedure [48]. HYPERDK and HYPERLAP also operate in this setting by preserving structural statistics over a fixed node set.

[1] We named our model **Janus** after the Roman god with two faces, symbolizing duality and transitions, which reflects the model's dual-view design.

3 Preliminaries

A hypergraph $\mathrm{H} = (\mathrm{V}, \mathrm{E})$ consists of a set of nodes V and a set of hyperedges E, where each hyperedge $e \in \mathrm{E}$ is a subset of nodes ($e \subseteq \mathrm{V}$). We represent it using an incidence matrix $\mathbf{H}$, whose entries satisfy $h_{ve} = 1$ if node v belongs to hyperedge e, and $h_{ve} = 0$ otherwise. The *dual hypergraph* exchanges the roles of nodes and hyperedges, corresponding to the transpose $\mathbf{H}^\top$. The *line graph* provides a graph-based view of the hypergraph in which hyperedges become nodes, and two such nodes are linked whenever the corresponding hyperedges overlap by sharing at least one node. Node and hyperedge attributes are stored in feature matrices $\mathbf{X}^{\mathrm{V}}$ and $\mathbf{X}^{\mathrm{E}}$.

Denoising Diffusion generates data by learning to reverse a predefined Gaussian noising process: $q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1}),\ \ q(x_t \mid x_{t-1}) = \mathcal{N}\big(x_t; \sqrt{\alpha_t}\, x_{t-1},\, \beta_t \mathbf{I}\big)$, where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. The noisy sample admits the closed form $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \varepsilon,\ \varepsilon \sim \mathcal{N}(0, I)$. For notational convenience, we define $a_t := \sqrt{\bar{\alpha}_t}$ and $\sigma_t := \sqrt{1 - \bar{\alpha}_t}$, so that $x_t = a_t x_0 + \sigma_t \varepsilon$. The model learns the reverse transitions, typically by predicting either the noise ε or the velocity $v_t = a_t \varepsilon - \sigma_t x_0$, which improves stability. From a velocity prediction $\hat{v}_\theta$, one can recover estimates of the clean data and noise:

$$\hat{x}_{0,\theta} = a_t x_t - \sigma_t \hat{v}_\theta(x_t, t), \qquad \hat{\varepsilon}_\theta = \sigma_t x_t + a_t \hat{v}_\theta(x_t, t). \tag{1}$$

Latent Diffusion Models reduce computational cost by applying diffusion in a latent space. An autoencoder (E, D) maps data to a representation $z = E(x)$ and reconstructs it via the decoder $D(z)$. The diffusion model operates on z, and generated samples are decoded to the data space.
Additional background and notation details are provided in App. A.

3.1 Problem formulation

We consider a single observed hypergraph $\mathrm{H} = (\mathrm{V}, \mathrm{E})$ and aim to learn the generative principles underlying its high-order interaction patterns. To this end, we model the global structure as the composition of smaller local configurations. We therefore introduce a stochastic sampling operator $\mathcal{S}(\mathrm{H}) \coloneqq \mathcal{H} = \{\mathrm{H}^s\}_{s=1}^{m}$, which extracts a collection of sampled sub-hypergraphs $\mathrm{H}^s = (\mathrm{V}^s, \mathrm{E}^s)$ from H. Each sampled sub-hypergraph H^s serves as a training instance.
This decomposition serves two main purposes. First, it transforms a single observed hypergraph into multiple structured training examples, providing a form of self-supervision. Second, it exposes the local statistical regularities that the generative model should learn. We therefore formulate the problem of hypergraph generation as the task of learning the distribution over sampled sub-hypergraphs induced by $\mathcal{S}$. Formally, we aim to approximate $p_{\mathrm{sample}}\left(\mathrm{H}^s \mid \mathrm{H}\right)$, using a parametric model p_θ, whose parameters are learned by solving $\theta^\star \in \arg\max_\theta \sum_{s=1}^{m} \log p_\theta\left(\mathrm{H}^s\right)$.
After training, the model generates sub-hypergraphs $\tilde{\mathrm{H}}^s \sim p_{\theta^\star}$, which are then assembled into a global hypergraph $\tilde{\mathrm{H}}$. Overall, this formulation reduces hypergraph generation to learning the distribution of sub-hypergraphs whose composition produces a coherent global structure. This perspective is especially suitable for the single-hypergraph setting: it improves scalability by operating on smaller structures and naturally captures the multi-scale organization of higher-order systems.

Node-set-constrained generation. In many applications, node identities are known a priori, while the interactions among them are unknown and need to be generated [5, 13]. We refer to this as the *node-set-constrained* setting, in contrast to the previously described *unconstrained* setting, where node identities are not fixed, even though their generation remains conditioned by the target number of nodes K used during entity resolution. Given an observed hypergraph $\mathrm{H} = (\mathrm{V}, \mathrm{E})$, we retain the node set V and compute the node representations $\mathbf{X}_{\mathrm{V}} = f(\mathrm{H})$ from the observed structure. The objective is to learn a conditional distribution $p_\theta(\tilde{\mathrm{E}} \mid \mathrm{V}, \mathbf{X}_{\mathrm{V}})$ over the hyperedge sets defined on the specified nodes, and to generate $\tilde{\mathrm{H}} = (\mathrm{V}, \tilde{\mathrm{E}})$ with $\tilde{\mathrm{E}} \sim p_{\theta^\star}(\cdot \mid \mathrm{V}, \mathbf{X}_{\mathrm{V}})$.
In our framework, this means modeling sampled sub-hypergraphs over subsets of V while keeping the node identities fixed. Since $\mathbf{X}_{\mathrm{V}}$ is derived from the observed hypergraph, this setting is transductive and structure-conditioned: the model generates alternative interaction structures over the prescribed entities while preserving structural information from the observed hypergraph, rather than performing topology-free generation conditioned only on V.

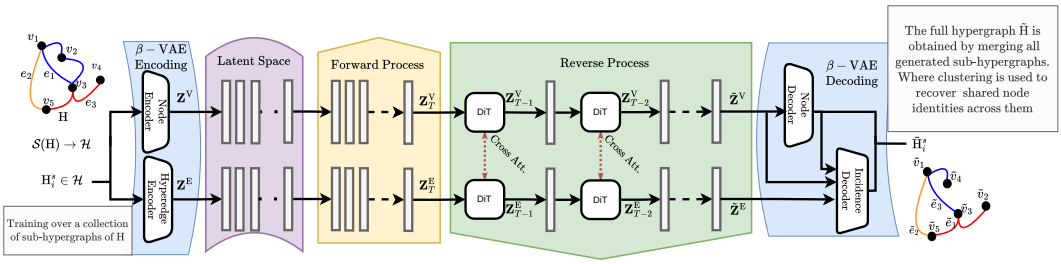


Figure 1: JANUS *at a glance*: our proposed hypergraph generator model and its core workflow.

4 Method

Figure 1 illustrates our latent diffusion model for hypergraph generation, JANUS, consisting of two components. First, we design a variant of the β-Variational Autoencoder (β-VAE) [20] tailored for hypergraphs, enabling structural reconstruction via joint latent representations of nodes and hyperedges, resulting in two latent spaces. Second, we introduce a continuous diffusion model operating in these latent spaces to permit the generation of new samples. Trained on representations of sampled sub-hypergraphs, the diffusion model generates latent samples that are decoded into valid sub-hypergraphs using the pretrained β-VAE decoder.

Accordingly, both the β-VAE and the diffusion model learn the induced distribution over these samples. Global generation is performed by sampling multiple sub-hypergraphs and merging them through an entity-resolution procedure that identifies node correspondences across sub-hypergraphs. Given a hypergraph $\mathrm{H} = (\mathrm{V}, \mathrm{E})$, we construct a collection $\mathcal{H}$ of sampled sub-hypergraphs by applying a sampler $\mathcal{S}$ to its line-graph representation. We sample one hypergraph $\mathrm{H}^s = (\mathrm{V}^s, \mathrm{E}^s) \in \mathcal{H}$ for each hyperedge in E, where each sample is obtained by selecting a subset of hyperedges $\mathrm{E}^s \subseteq \mathrm{E}$ and defining V^s as the union of the nodes incident to the hyperedges in E^s. In practice, we use a random sampling strategy, and to encourage connectivity, we employ a biased random walk [18] with a Metropolis–Hastings correction [19]. This procedure defines the training distribution and underpins both representation learning and latent diffusion. For the entity resolution procedure, we merge the generated sub-hypergraphs by applying K-means clustering to their node representations, providing K as input to JANUS. Like the other unconstrained baselines considered in this paper, JANUS does not condition on individual node or hyperedge features at generation time and can, therefore, generate hypergraphs with a user-specified target number of nodes and hyperedges.

The following sections describe both components and the neural network configurations used for learning. We report complete implementation details in App. B.

4.1 Dual-view β-VAE for hypergraph learning

Our model builds on the β-VAE framework and adapts it to hypergraph representation learning by introducing a dual-view variational architecture. The model employs a joint encoder-decoder structure that learns two interacting latent spaces corresponding to nodes and hyperedges. Unlike standard β-VAEs, which operate on a single latent representation, our approach explicitly models the bipartite incidence structure of hypergraphs via separate but coupled latent variables for nodes and hyperedges, both of dimension d_z in our setting.

Let $\mathbf{X}^{\mathrm{V}} \in \mathbb{R}^{N \times d_{\mathrm{V}}}$ and $\mathbf{X}^{\mathrm{E}} \in \mathbb{R}^{M \times d_{\mathrm{E}}}$ denote node and hyperedge features of dimensions d_{V} and d_{E}, respectively, and let $\mathbf{H} \in \{0,1\}^{N \times M}$ denote the incidence matrix. We introduce two families of latent variables: $\mathbf{Z}^{\mathrm{V}} = [\mathbf{z}_1^{\mathrm{V}}, \ldots, \mathbf{z}_N^{\mathrm{V}}] \in \mathbb{R}^{N \times d_z}$, $\mathbf{Z}^{\mathrm{E}} = [\mathbf{z}_1^{\mathrm{E}}, \ldots, \mathbf{z}_M^{\mathrm{E}}] \in \mathbb{R}^{M \times d_z}$, where $\mathbf{z}_i^{\mathrm{V}}$ is the latent representation of node i, and $\mathbf{z}_j^{\mathrm{E}}$ is the latent representation of hyperedge j.

Node and hyperedge encoders. Following the variational autoencoder formulation, we place independent isotropic Gaussian priors on the node and hyperedge latent variables: $p(\mathbf{Z}^{\mathrm{V}}) = \prod_{i=1}^{N} \mathcal{N}\big(\mathbf{z}_i^{\mathrm{V}} \mid \mathbf{0}, \mathbf{I}_{d_z}\big)$ and $p(\mathbf{Z}^{\mathrm{E}}) = \prod_{j=1}^{M} \mathcal{N}\big(\mathbf{z}_j^{\mathrm{E}} \mid \mathbf{0}, \mathbf{I}_{d_z}\big)$. We then approximate the posterior distribution with a variational distribution that factorizes across the two views:

$$q_\phi\big(\mathbf{Z}^{\mathrm{V}}, \mathbf{Z}^{\mathrm{E}} \mid \mathbf{X}^{\mathrm{V}}, \mathbf{X}^{\mathrm{E}}, \mathbf{H}\big) = \underbrace{q_{\phi_{\mathrm{V}}}\big(\mathbf{Z}^{\mathrm{V}} \mid \mathbf{X}^{\mathrm{V}}, \mathbf{H}\big)}_{\text{node encoder}} \underbrace{q_{\phi_{\mathrm{E}}}\big(\mathbf{Z}^{\mathrm{E}} \mid \mathbf{X}^{\mathrm{E}}, \mathbf{H}^{\top}\big)}_{\text{hyperedge encoder}}.$$

Each encoder is modeled as: $q_{\phi_{\mathrm{V}}}\big(\mathbf{Z}^{\mathrm{V}} \mid \mathbf{X}^{\mathrm{V}}, \mathbf{H}\big) = \prod_{i=1}^{N} \mathcal{N}\big(\mathbf{z}_i^{\mathrm{V}} \mid \boldsymbol{\mu}_i^{\mathrm{V}}, \mathrm{diag}\big((\boldsymbol{\sigma}_i^{\mathrm{V}})^2\big)\big)$ and $q_{\phi_{\mathrm{E}}}\big(\mathbf{Z}^{\mathrm{E}} \mid \mathbf{X}^{\mathrm{E}}, \mathbf{H}^{\top}\big) = \prod_{j=1}^{M} \mathcal{N}\big(\mathbf{z}_j^{\mathrm{E}} \mid \boldsymbol{\mu}_j^{\mathrm{E}}, \mathrm{diag}\big((\boldsymbol{\sigma}_j^{\mathrm{E}})^2\big)\big)$. The two encoders estimate the parameters $\{\boldsymbol{\mu}_i^{\mathrm{V}}, \boldsymbol{\sigma}_i^{\mathrm{V}}\}_{i=1}^{N}$ and $\{\boldsymbol{\mu}_j^{\mathrm{E}}, \boldsymbol{\sigma}_j^{\mathrm{E}}\}_{j=1}^{M}$, respectively, by learning neural-network functions defined on the hypergraph and on its dual. Specifically, both encoders adopt a Hypergraph Attention Layer [6] with skip connections and a transformer-like attention mechanism [46].

Incidence decoder. This component generates the incidence matrix $\mathbf{H}$ from the interaction between node and hyperedge latent embeddings:

$$p_\theta\big(\mathbf{H}, \mathbf{Z}^{\mathrm{V}}, \mathbf{Z}^{\mathrm{E}}\big) = p(\mathbf{Z}^{\mathrm{V}})\, p(\mathbf{Z}^{\mathrm{E}})\, \underbrace{p_\theta\big(\mathbf{H} \mid \mathbf{Z}^{\mathrm{V}}, \mathbf{Z}^{\mathrm{E}}\big)}_{\text{incidence decoder}}. \tag{2}$$

Under the assumption that the incidence entries are conditionally independent given the node and hyperedge latent embeddings, the likelihood factorizes as $p_\theta\big(\mathbf{H} \mid \mathbf{Z}^{\mathrm{V}}, \mathbf{Z}^{\mathrm{E}}\big) = \prod_{i=1}^{N} \prod_{j=1}^{M} p_\theta\big(h_{ij} \mid \mathbf{z}_i^{\mathrm{V}}, \mathbf{z}_j^{\mathrm{E}}\big)$. This assumption is analogous to the edge-independent decoders commonly used in graph variational autoencoders [27].

The node and hyperedge embeddings are combined through an element-wise product, followed by LayerNorm, and then linearly projected onto a two-dimensional output space representing the binary incidence outcome: $p_\theta\big(h_{ij} = c \mid \mathbf{z}_i^{\mathrm{V}}, \mathbf{z}_j^{\mathrm{E}}\big) = \mathrm{softmax}\big(\mathbf{W}_\theta\ \mathrm{LN}\big(\mathbf{z}_i^{\mathrm{V}} \odot \mathbf{z}_j^{\mathrm{E}}\big) + \mathbf{b}_\theta\big)_c$ with $c \in \{0, 1\}$, where $\mathbf{W}_\theta \in \mathbb{R}^{2 \times d_z}$ and $\mathbf{b}_\theta \in \mathbb{R}^2$ are the projection matrix and bias vector. At sampling time, we sample an incidence matrix from the decoder: $\tilde{\mathbf{H}} \sim p_\theta\big(\mathbf{H} \mid \mathbf{Z}^{\mathrm{V}}, \mathbf{Z}^{\mathrm{E}}\big)$.

Entity resolution node decoder. For a sampled sub-hypergraph H^s, we define a deterministic node decoder $\tilde{\mathbf{X}}_s^{\mathrm{V}} = g_\omega\big(\mathrm{H}^s, \mathbf{Z}_s^{\mathrm{V}}, \mathbf{Z}_s^{\mathrm{E}}\big) \in \mathbb{R}^{|\mathrm{V}^s| \times d_c}$(3). We denote by $\tilde{\mathbf{x}}_{u,s}^{\mathrm{V}} \in \mathbb{R}^{d_c}$ the decoded representation associated with node occurrence u in sample s, where d_c is the decoder output dimension. Since sampled sub-hypergraphs may overlap, the same underlying node entity can appear in multiple samples. *Therefore, to enable entity resolution, we encourage decoded representations corresponding to the same node across different sub-hypergraphs to occupy nearby regions of a shared representation space.* For this reason, we exploit supervision during training by forming positive pairs $\mathcal{P}$ from repeated occurrences of the same original observed node in different sub-hypergraphs, while using occurrences of other nodes as negatives. The decoder is trained with a contrastive InfoNCE objective:

$$\mathcal{L}_{\mathrm{con}} = -\frac{1}{|\mathcal{P}|} \sum_{((u,a),(u,b)) \in \mathcal{P}} \log \frac{\exp\Big(\mathrm{sim}\Big(\tilde{\mathbf{x}}_{u,a}^{\mathrm{V}}, \tilde{\mathbf{x}}_{u,b}^{\mathrm{V}}\Big)/\tau\Big)}{\sum_{(c,d) \in \mathcal{B}} \exp\Big(\mathrm{sim}\Big(\tilde{\mathbf{x}}_{u,a}^{\mathrm{V}}, \tilde{\mathbf{x}}_{c,d}^{\mathrm{V}}\Big)/\tau\Big)}, \qquad \mathrm{sim}(\mathbf{p}, \mathbf{q}) = \frac{\mathbf{p}^{\top}\mathbf{q}}{\|\mathbf{p}\|\,\|\mathbf{q}\|},$$

where $\mathcal{B}$ is the set of node occurrences in the current batch and $\tau > 0$ is a temperature parameter, and sim is the cosine similarity function. During training, we use teacher forcing and compute the decoder outputs from the ground-truth incidence matrix associated with H^s, rather than from the decoded incidence matrix $\tilde{\mathbf{H}}^s$, which stabilizes optimization. As introduced above, our implementation adopts K-means for entity resolution; accordingly, the training objective encourages embeddings corresponding to the same entity to fall into the same cluster.

4.2 Dual-view hypergraph latent diffusion

We define the latent diffusion model over coupled embeddings $\mathbf{Z}_0^{\mathrm{V}} \in \mathbb{R}^{N \times d_z}, \mathbf{Z}_0^{\mathrm{E}} \in \mathbb{R}^{M \times d_z}$, which we interpret as *clean representations* of nodes and hyperedges that are free of noise. During training, the clean pair $(\mathbf{Z}_0^{\mathrm{V}}, \mathbf{Z}_0^{\mathrm{E}})$ is sampled from the β-VAE approximate posterior $\sim q_\phi(\mathbf{Z}^{\mathrm{V}}, \mathbf{Z}^{\mathrm{E}} \mid \mathbf{X}^{\mathrm{V}}, \mathbf{X}^{\mathrm{E}}, \mathbf{H})$. The diffusion model is trained to learn the latent distribution induced by sampled sub-hypergraphs, so that it can generate new paired latents that can be decoded into sub-hypergraphs.

Forward noising process. To corrupt the clean latents, we run two *independent* forward diffusion processes: one on the node latent representations and one on the hyperedge latent representations. Each process is a Markov chain that gradually adds noise over T steps: $q(\mathbf{Z}_{1:T}^{\mathrm{V}} \mid \mathbf{Z}_0^{\mathrm{V}}) = \prod_{t=1}^{T} q(\mathbf{Z}_t^{\mathrm{V}} \mid \mathbf{Z}_{t-1}^{\mathrm{V}})$, and $q(\mathbf{Z}_{1:T}^{\mathrm{E}} \mid \mathbf{Z}_0^{\mathrm{E}}) = \prod_{t=1}^{T} q(\mathbf{Z}_t^{\mathrm{E}} \mid \mathbf{Z}_{t-1}^{\mathrm{E}})$. Here, $\mathbf{Z}_t^{\mathrm{V}}$ and $\mathbf{Z}_t^{\mathrm{E}}$ are the progressively noised representations at diffusion step t. For the remainder of the forward process formulation, we follow the denoising diffusion framework introduced in Sec. 3.

Reverse denoising process. We model the reverse diffusion process using two denoisers: one for

node latents and one for hyperedge latents. Both denoisers use the same architecture, but have distinct parameter sets. Although the node and hyperedge latent spaces represent different types of objects, their generation must be modeled jointly, as the decoder reconstructs the incidence matrix from the interactions between node and hyperedge embeddings. As a result, the fidelity of the generated hypergraph depends not only on the plausibility of each latent representation individually, but also on their mutual consistency. Denoising the two streams independently would allow each denoiser to learn the distribution of its respective latent space, without explicitly accounting for the relationships between the two views. To encourage cross-view consistency, we couple the two reverse processes through cross-attention at each step, allowing the node and hyperedge denoisers to condition on each other's current latent representations. Formally, the joint reverse chain factorizes as

$$p_{\psi,\phi}\big(\mathbf{Z}^{\mathrm{V}}_{0:T},\, \mathbf{Z}^{\mathrm{E}}_{0:T}\big) = p\big(\mathbf{Z}^{\mathrm{V}}_{T}\big)\; p\big(\mathbf{Z}^{\mathrm{E}}_{T}\big) \prod_{t=T}^{1} p_{\psi}\big(\mathbf{Z}^{\mathrm{V}}_{t-1} \mid \mathbf{Z}^{\mathrm{V}}_{t},\, \mathbf{Z}^{\mathrm{E}}_{t}\big) \prod_{t=T}^{1} p_{\phi}\big(\mathbf{Z}^{\mathrm{E}}_{t-1} \mid \mathbf{Z}^{\mathrm{E}}_{t},\, \mathbf{Z}^{\mathrm{V}}_{t}\big)\,. \quad (4)$$

Our denoising networks follow the formulation in Sec. 3. Specifically, we use the v-prediction parameterization defined in Eq. 1 and instantiate two denoisers, one for node latents and one for hyperedge latents. We implement each denoiser as a Diffusion Transformer [40]. We condition both networks on the diffusion timestep through adaptive layer normalization (adaLN). To couple the two latent streams, we introduce cross-attention: the node denoiser v_ψ denoises node latents while attending to the current hyperedge latents, whereas the hyperedge denoiser v_ϕ denoises hyperedge latents while attending to the current node latents.

Node-set constraint. In this setting, we omit the node decoder because the model receives the original node representations as input and generates only hyperedges; consequently, it does not require diffusion over node latents. Note that these representations are processed through a multilayer perceptron before decoding. We refer to the unconstrained version of our method as JANUS, and to the node-set-constrained variant as JANUSNC.

4.3 Training strategy

We train the hypergraph β-VAE by *alternating* between two objectives: incidence reconstruction and entity resolution. During each training epoch, gradients flow only through the component associated with the active objective, while the remaining component stays fixed. We first optimize the incidence decoder by maximizing the β-weighted ELBO, which combines an incidence-matrix reconstruction term with a weighted Kullback–Leibler (KL) regularization term:

$$\mathcal{L}_\beta = \mathbb{E}\Big[\; \underbrace{\log p_\theta(\mathbf{H} \mid \mathbf{Z}^{\mathrm{V}}, \mathbf{Z}^{\mathrm{E}})}_{\text{Incidence-matrix reconstruction}} \; \underbrace{-\beta\Big(\mathrm{KL}\big(q_{\phi_{\mathrm{V}}}(\mathbf{Z}^{\mathrm{V}} \mid \mathbf{X}^{\mathrm{V}}, \mathbf{H}) \,\|\, p(\mathbf{Z}^{\mathrm{V}})\big) + \mathrm{KL}\big(q_{\phi_{\mathrm{E}}}(\mathbf{Z}^{\mathrm{E}} \mid \mathbf{X}^{\mathrm{E}}, \mathbf{H}^\top) \,\|\, p(\mathbf{Z}^{\mathrm{E}})\big)\Big)}_{\mathcal{L}_{\mathrm{KL}}\text{ (KL divergence)}} \Big].$$

As shown in Fig. 1, the incidence reconstruction loss updates both encoders and the incidence decoder. In the next epoch, we optimize the node decoder for entity resolution using the contrastive loss $\mathcal{L}_{\mathrm{con}}$, which updates only the node encoder and the node decoder. For latent diffusion, we train the two denoisers by minimizing the following regression objective:

$$\mathcal{L}_{\mathrm{diff}} = \mathbb{E}_{(\mathbf{Z}^{\mathrm{V}}, \mathbf{Z}^{\mathrm{E}}) \sim q_\phi,\, t,\, \boldsymbol{\epsilon}^{\mathrm{V}},\, \boldsymbol{\epsilon}^{\mathrm{E}}} \Big[\big\| \mathbf{v}^{\mathrm{V}}_t - \hat{\mathbf{v}}_\psi(\mathbf{Z}^{\mathrm{V}}_t, \mathbf{Z}^{\mathrm{E}}_t, t) \big\|_F^2 + \big\| \mathbf{v}^{\mathrm{E}}_t - \hat{\mathbf{v}}_\phi(\mathbf{Z}^{\mathrm{E}}_t, \mathbf{Z}^{\mathrm{V}}_t, t) \big\|_F^2 \Big].$$

Following the v-prediction formulation described in Sec. 3, we train the denoisers to regress the target velocity. At each iteration, we sample a timestep $t \sim \mathrm{Unif}(\{1, \ldots, T\})$ and draw independent Gaussian noises $\boldsymbol{\epsilon}^{\mathrm{V}}, \boldsymbol{\epsilon}^{\mathrm{E}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. We use a linear variance schedule with $T = 1000$ timesteps. We optimize both networks with *AdamW* and, following the original *Diffusion Transformer* protocol, set the weight decay to zero. The β-VAE and the latent diffusion model are trained as separate processes.

4.4 Hypergraph generation procedure

JANUS generates hypergraphs through a two-stage pipeline: it first generates sub-hypergraphs and then merges them to obtain the full hypergraph. In the first stage, we sample Gaussian noise and apply the reverse diffusion process in Eq. 4 to obtain clean node and hyperedge latents, $(\tilde{\mathbf{Z}}^{\mathrm{V}}_0, \tilde{\mathbf{Z}}^{\mathrm{E}}_0) \sim p_{\psi,\phi}$. The β-VAE incidence decoder in Eq. 2 maps these latents to a generated incidence matrix. In contrast, the entity-resolution node decoder in Eq. 3 produces node representations used to identify recurring node entities across generated samples. We then retain only non-empty and unique hyperedges, yielding a

valid generated sub-hypergraph $\tilde{\mathrm{H}}^s$. During inference, we apply this procedure to generate a collection of sub-hypergraphs $\tilde{\mathcal{H}}$, and merge them into the global hypergraph (this can be performed in one shot or until the generated network has a desired size). We then apply the entity-resolution procedure described above, based on K-means clustering, to identify node occurrences that correspond to the same underlying entity across different sub-hypergraphs. This step allows us to merge the generated sub-hypergraphs into a single full hypergraph. The **computational complexity** of JANUS is $O((M + ST)(n + m)^2 + SnN)$ for the unconstrained variant and $O((M + ST)(n + m)^2)$ for JANUSNC, where M denotes the number of hyperedges in the original hypergraph; n and m denote the numbers of nodes and hyperedges, respectively, in a sampled sub-hypergraph; S denotes the number of generated sub-hypergraphs; and T denotes the number of diffusion steps. Further details on the complexity analysis are provided in Appendix H.1.

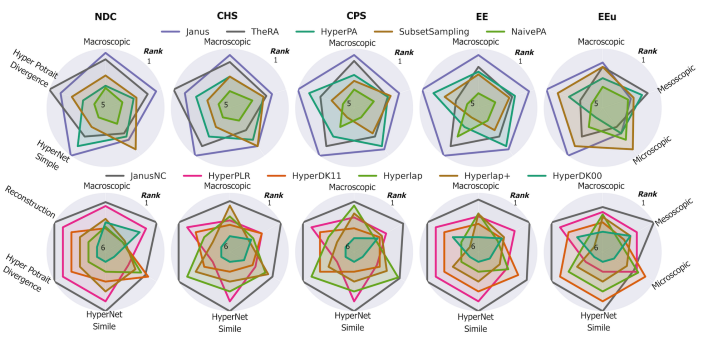
5 Experiments

We designed a novel hypergraph-specific **evaluation framework** to assess generative model performance across multiple incremental structural scales, enabling more reproducible and fair comparisons across methods. Specifically, at the *microscopic level*, we compare the node degree distribution (*NDD*, Wasserstein distance between real and generated), the hyperedge size distribution (*HSD*), and the clique-expansion clustering coefficient (*CECC*); at the *mesoscopic level*, we quantify high-order organization using community recovery with the NMI [45], degree assortativity (*DA*) [38], and closed/open triangle (*C/O-T*) statistics computed on the clique expansion; at the *macroscopic level*, we measure incidence density (*ID*), the diameter of the largest connected component (*LCCD*), hypergraph transitivity (*HT*) [25], and global and egonet overlapness (*GO*/*EO*) [32]. At the macroscopic scale, we also match dataset statistics using node and hyperedge counts (*N/H-C*) and projection-based edge counts on the bipartite, line graph, and clique expansions (*B/L/C-PEC*). To characterize global community structure, we report modularity under four complementary views: clique expansion, line graph, node–hyperedge bipartite representation, and hypergraph (*C/L/BP/H-M*) [30]. Further, we assess whether generated samples are empirically indistinguishable from real hypergraphs using the following high-order dissimilarities: *Hyper NetSimile* (*HNS*) and *Hyperedge Portrait Divergence* (*HPD*) [1]. Finally, in the node-set-constrained setting, we additionally evaluate *reconstruction* by measuring the number of correctly recovered hyperedges (*HR*) under the prescribed node set and the Jaccard coefficient (*JH*) between the real and generated hyperedges. Details are reported in App. E. Experiments were performed on a machine with 24 CPUs, 188 GB of RAM, and an NVIDIA RTX 5090 GPU with 32 GB of memory; training and generation times for EE are reported in App. H.

Baselines. We compare JANUS against representative hypergraph generators that reflect different modeling assumptions and provide different structural guarantees. In the unconstrained setting, we include HYPERPA, SUBSETSAMPLING, and NAIVEPA as preferential-attachment-based generators, and THERA as a transitivity-oriented method. In the constrained node-set setting (Sec. 3.1), where the node set is fixed and provided as input, we include HYPERPLR, a projection–learning–reconstruction method, HYPERDK00 and HYPERDK11 as local-statistics-preserving randomization models, and HYPERLAP/HYPERLAP+ as overlapness-oriented generators.

We exclude DGMH as a baseline because no public implementation is available. Nevertheless, for completeness, in Sec. 5.2, we reproduce their experiments on two datasets for JANUSNC. We do not include HYGENE as a baseline because it addresses a different generation setting since it learns from a collection of independent, fully observed hypergraphs, whereas JANUS operates on a single large hypergraph by sampling overlapping local views and merging generated samples into a coherent global structure. Non-overlapping partitions would not reproduce the JANUS setting, as they would either discard interactions across partition boundaries or produce disconnected components. JANUS instead relies on overlapping samples to preserve boundary information and repeated node identities, which are reconciled through entity resolution. Adapting HYGENE to our setting would therefore require substantial changes, including overlapping-view sampling and shared-entity resolution. We consequently treat the two methods as addressing different, not directly comparable, tasks.

Datasets. We evaluated JANUS on five real-world hypergraph datasets from [7, 49, 33]. For each dataset, we report the number of nodes and hyperedges in parentheses: *email-Enron*, EE (143, 1512), *email-Eu*, EEu (998, 25027), *contact-high-school* [36], CHS (327, 7818), *contact-primary-school* [44], CPS (242, 12704), and NDC-classes, NDC (1161, 1088). We provide additional dataset details in App. D. We computed node features using `node2vec` [18], and we removed duplicate hyperedges.



Dataset / Metric	1st	2nd	3rd
	unconstrained		
NDC	JANUS	THERA	SUBSETSAMPLING
HNS	6.04 ±0.30	9.77 ±0.00	9.89 ±0.47
CHS	JANUS	THERA	HYPERPA
NDD	2.95 ±0.49	13.51 ±0.00	31.47 ±1.57
CPS	JANUS	THERA	HYPERPA
NDD	5.56 ±0.15	22.99 ±0.00	67.07 ±0.63
EE	JANUS	THERA	HYPERPA
NDD	3.53 ±0.37	10.63 ±0.00	11.70 ±1.38
EEu	SUBSETSAMPLING	THERA	JANUS
NDD	30.56 ±5.39	62.44 ±0.00	64.06 ±0.74
	node-set-constrained		
NDC	JANUSNC	HYPERPLR	HYPERDK11
HNS	2.56 ±0.06	3.35 ±0.11	8.55 ±0.05
CHS	JANUSNC	HYPERLAP+	HYPERPLR
NDD	0.73 ±0.06	1.22 ±0.12	111.12 ±0.23
CPS	JANUSNC	HYPERLAP	HYPERPLR
NDD	2.11 ±0.09	2.29 ±0.39	16.00 ±0.47
EE	JANUSNC	HYPERPLR	HYPERDK11
NDD	0.85 ±0.11	1.58 ±0.11	3.23 ±0.18
EEu	HYPERDK11	HYPERPLR	JANUSNC
NDD	3.00 ±0.11	6.43 ±0.57	23.59 ±0.50

Figure 2: **Results highlights.** *Left*: average ranks for unconstrained (top row) and node-set-constrained methods (bottom row) across all metric: micro-, meso-, and macro-scale metrics, *HNS*, *HPD*, and reconstruction. *Right*: most challenging metrics among the top-three methods.

5.1 Results

We train the β-VAE until convergence, using at least 3000 epochs, using $\beta = 1e-6$ as for [41]. In generative modeling, sample-quality metrics are commonly used for checkpoint selection because training losses are often weakly correlated with final sample quality [39, 12, 23]. Specifically, we empirically use high-order dissimilarity measures as indicators of generation quality, analogous to FID-based checkpointing in image generation [24].[2] Importantly, these metrics are used only on validation-generated samples for checkpoint selection. The final results are computed independently after checkpoint selection, and we do not tune the model architecture or hyperparameters against the final evaluation scores. To provide a compact yet comprehensive comparison across all evaluation metrics, we summarize performance using the average rank of each method over datasets. For each metric, we average the results over 5 generated hypergraphs, following prior work [48]. We group the metrics according to the evaluation categories described in the previous section. Figure 2 summarizes these results in the left panel; App. F reports the full numerical values for each metric. We observed that the Jaccard overlap between the resulting training and validation sub-hypergraphs ranges from 0.0156% to 0.2119% across datasets; details are reported in App. C.1.

Overall. JANUS shows the most consistent performance across datasets and structural scales. In the size-conditioned setting, it obtains the best aggregated rank on NDC, CHS, CPS, and EE, and ranks second on EEu, where SUBSETSAMPLING obtains an average rank of 2.1 compared with 2.3 for JANUS. In particular, JANUS obtains the best macroscopic rank on every dataset and ranks first in *HNS* throughout. It also achieves the best mesoscopic rank on all datasets except EEu, and the best *HPD* results on CPS, EE, and EEu. In the fixed-node, structure-conditioned setting, JANUSNC remains more consistent: it obtains the best overall aggregated rank on all five datasets and ranks first in every evaluation category except the microscopic category on NDC and EEu. The main gains therefore span several structural scales rather than concentrating in a single family of metrics, with the clearest improvements at the mesoscopic level, in high-order dissimilarity, and in reconstruction.

Microscopic highlight. At the local scale, JANUS remains competitive, but the gap with baselines is smaller. In the unconstrained setting, JANUS is best on CPS and EE, ties with SUBSETSAMPLING on CHS, and is outperformed by SUBSETSAMPLING on NDC and EEu. In the node-set-constrained setting, JANUSNC is best on CHS, CPS, and EE, while HYPERDK11 is best on NDC and EEu. The strongest gains are: on CPS, JANUSNC reduces the *CECC* error from 0.0292 ± 0.0009 for HYPERLAP to 0.0038 ± 0.0015; on EE, JANUS reduces the same error from 0.0734 ± 0.0403 for HYPERPA to 0.0112 ± 0.0054; and on CHS, JANUS reduces the NDD distance from 13.51 ± 0 for THERA to 2.95 ± 0.4882.

Mesoscopic highlight. The clearest separation occurs at the mesoscopic level. In the unconstrained setting, JANUS ranks first on all datasets except EEu, where THERA and HYPERPA rank best. In the node-set-constrained setting, JANUSNC ranks first on all datasets. The improvements are particularly pronounced for triangle-based statistics. For example, on NDC, JANUSNC reduces the *O-T* count error from 3509 ± 280.34 for HYPERDK00 to 11.6 ± 14.29. This suggests that the proposed model

[2] FID, or Fréchet Inception Distance, is a standard sample-quality metric in image generation that compares the distributions of real and generated samples in a learned feature space.

is particularly effective at preserving intermediate-scale organization involving communities and higher-order motifs.

Macroscopic highlight. At the global scale, JANUS and JANUSNC rank first across all datasets, indicating strong agreement with global properties. The largest macroscopic gaps include the *LCCD* error on CHS in the node-set-constrained setting, where JANUSNC matches the value of the real dataset (with ± 0); the *GO* error on CPS in the unconstrained setting, where JANUS obtains 0.33 ± 0.17 versus 10.12 ± 6.18 for HYPERPA, the second-ranked method; and the *HM* error on EE in the unconstrained setting, where JANUS obtains 0.0047 ± 0.0039 versus 0.1377 ± 0 for THERA. Among the baselines, THERA is usually the strongest macroscopic competitor, reflecting the benefit of explicitly modeling transitivity. Preferential-attachment methods such as HYPERPA, NAIVEPA, and SUBSETSAMPLING can capture heavy-tailed or growth-based patterns, but they do not consistently preserve global organization.

High-order dissimilarity highlight. JANUS ranks first in *HNS* on every dataset, and JANUSNC ranks first in both *HNS* and *HPD* on every dataset. For instance, on CHS, JANUSNC reduces *HNS* from 2.07 ± 0.0146 for HYPERPLR to 0.19 ± 0.017. In the size-conditioned setting, the results are more mixed for *HPD*: JANUS ranks first on CPS, EE, and EEu, whereas THERA ranks first on NDC and CHS. Thus, while transitivity-oriented methods can better match specific high-order structural properties on some datasets, JANUS remains consistently strong across the two high-order dissimilarity measures.

Reconstruction highlight. JANUSNC is the strongest reconstruction method. It ranks first in reconstruction on all datasets, while HYPERPLR is consistently second. The largest gains are observed in *HR*. On EEu, JANUSNC increases *JH* from 0.0705 ± 0.002 for HYPERPLR to 0.4155 ± 0.0016. On EE, it increases *JH* from 0.1613 ± 0.0085 to 0.9215 ± 0.006. On EEu, it also increases *HR* from 3297 ± 85.50 to 14693 ± 38.76. This confirms that HYPERPLR effectively recovers observed hyperedges but is less reliable at preserving full structure across all categories.

5.2 JanusNC vs. DGMH highlight

Following the k-level decomposition used in the DGMH paper, we evaluate the NDC-classes and email-Eu hypergraphs through node (n), edge (e), triangle (t), and 4-clique (4c) views, using their reported metrics: the clustering coefficient, the largest connected component ratio, and the diameter. Detailed views, metrics, and results are reported in App. F.9. Compared with DGMH , JANUSNC gives the closest overall match to the real hypergraphs, mainly because it preserves diameter more accurately across both datasets. On email-Eu, JANUSNC nearly matches the real node, edge, triangle, and 4-clique diameters, with values of 2.82, 5.73, 10.63, and 16.59, compared with real values of 2.82, 5.71, 10.40, and 14.90, whereas DGMH is less accurate, with values of 1.67, 5.48, 9.51, and 18.62. On NDC-classes, JANUSNC is also closer than DGMH across all views, with values of 4.50, 6.58, 6.28, and 2.70 compared with real values of 4.45, 6.62, 6.51, and 2.69, although the difference is less pronounced. DGMH is competitive in selected local cases, such as email-Eu higher-order clustering 0.83 (t), 0.91 (4c) and connected components 0.77 (e), 0.76 (t), 0.42 (4c), which are close to the corresponding real values 0.82, 0.89 and 0.70, 0.80, 0.41. However, JANUSNC is more reliable on NDC-classes, especially for node- and edge-level connected components, where it obtains 0.60 (n), 0.62 (e) compared with real values of 0.54, 0.58, while DGMH overestimates them as 0.73 (n), 0.79 (e). Overall, DGMH captures some higher-order local patterns, but JANUSNC provides a more consistent reconstruction of both local and global structure.

5.3 Ablation study

We conducted an ablation study to assess the contribution of four design choices: entity resolution, InfoNCE, cross-attention, and latent diffusion. Full results and additional ablations are provided in App. G.

Role of diffusion vs. clustering-based reconstruction. To evaluate the role of the diffusion-based node representations, we use the node representations produced by the β-VAE encoder to assess both the decoder and the K-means reconstruction capability. Since these representations are the most favorable inputs to the decoder, this setting provides an upper bound on node-identity reconstruction. The ablation shows that entity resolution is effective but dataset-dependent: CHS, CPS, EE, and EEu achieve ARI, NMI, and AMI values close to one, whereas NDC is lower, with ARI $= 0.50184$, NMI

$= 0.88380$, and AMI $= 0.86824$; see App. G.1.[3] This suggests that fragmentation, as in NDC with 183 connected components, makes local-to-global entity resolution more ambiguous and can limit reconstruction quality.

Removing InfoNCE. Training the model without the contrastive loss degrades entity consistency and structural fidelity: on email-Enron, InfoNCE reduces HNS and HPD errors by approximately 58% and 44%, respectively. This supports the role of the contrastive objective in aligning repeated occurrences of the same underlying node across overlapping sub-hypergraphs. Full results are reported in App. G.2.

Effect of cross-attention in paired diffusion. We compare JANUS with an ablated variant that removes cross-attention between the node and hyperedge diffusion streams. Here, we report the high-order dissimilarity metrics, while the additional metrics, together with the full numerical results, are reported in App. G.3. Removing this coupling worsens the key high-order dissimilarity metrics: Hyper Portrait Divergence increases from 0.1158 to 0.1924, and Hyper NetSimile increases from 0.8984 to 1.6320. Hypergraph modularity also degrades from 0.0047 to 0.0133. The results show that cross-attention helps align node and hyperedge latent representations during denoising, producing more coherent incidence structures and high-order fidelity. Overall, considering all metrics, the cross-attention variant achieves an average improvement of approx. 50% over the ablated model.

VAE-only. We replace latent diffusion with direct sampling from an isotropic Gaussian prior while retaining the same decoding pipeline. On email-Enron, JANUS improves over this VAE-only baseline across all evaluated structural scales. Error reductions range from approximately 36% for the line-graph edge count to nearly 100% for open triangles, while HNS and HPD improve by approximately 89% and 85%, respectively. Thus, good VAE reconstruction alone does not ensure that direct prior samples correspond to realistic hypergraphs; latent diffusion is needed to model structurally meaningful latent configurations. Complete results, including the fixed-node, structure-conditioned setting, are reported in App. G.4.

6 Conclusion

In this work, we presented JANUS, a latent diffusion framework for hypergraph generation that models nodes and hyperedges as coupled latent processes. By combining a dual-view β-VAE with cross-conditioned diffusion, JANUS generates coherent incidence structures without applying diffusion directly to the discrete hypergraph structure. A key contribution of this work is a novel hypergraph-specific evaluation framework designed to assess generative performance across multiple scales, enabling more reproducible and fair comparisons across methods. Using this framework, experiments across five real-world datasets show that JANUS is competitive across all scales and particularly effective in the node-set-constrained setting, where JANUSNC achieves the strongest reconstruction performance. These results suggest that coupled latent diffusion is a promising direction for high-order relational generation. Finally, as a broader impact, JANUS may help researchers study sensitive high-order data without directly sharing the original data; however, generated hypergraphs may still pose privacy or misuse risks without appropriate safeguards.

Limitations and future work. JANUS relies on a sampling operator to construct sub-hypergraphs. We mainly use biased random-walk sampling and also test uniform random sampling, observing no significant performance differences. Exploring other sampling strategies is left for future work.

In the node-set-constrained setting, JANUSNC follows a transductive protocol, consistent with projection-based methods such as HYPERPLR. Its `node2vec` features are computed on the clique expansion of the observed hypergraph, and therefore encode pairwise co-occurrence information from the original hyperedges. Thus, JANUSNC results should be interpreted as structure-conditioned reconstruction/generation over a fixed node set, rather than inductive generation from topology-free node attributes. Future work will investigate the integration of classifier-free guidance, exploring whether JANUS can generate hypergraphs that follow the original distribution while also incorporating novel and more complex properties at different structural scales.

[3] Normalized Mutual Information (NMI): Shared information between two clusterings; higher is better. Adjusted Mutual Information (AMI): Mutual information adjusted for chance; 1 is perfect, 0 is random. Adjusted Rand Index (ARI): Pairwise clustering agreement adjusted for chance; 1 is perfect, 0 is random, and -0.5 is poor performance.

Acknowledgments and Disclosure of Funding

This work has been partially supported by the IT4LIA AI FACTORY project (co-funded by the Italian Ministry of University and Research, the Italian National Cybersecurity Agency, and the European Commission) through the European High Performance Joint Undertaking under Grant Agreement No. 101234224.

Competing interests. The authors declare no competing interests.

The authors Alessia Antelmi and Mirko Polato dedicate this work to Prof. Robert Birke, *who will always inspire us to be the best version of ourselves*.

References

[1] Cosimo Agostinelli, Marco Mancastroppa, and Alain Barrat. Higher-order dissimilarity measures for hypergraph comparison. *Journal of Complex Networks*, 14(1):cnaf048, 2026.

[2] Alessia Antelmi, Gennaro Cordasco, Mirko Polato, Vittorio Scarano, Carmine Spagnuolo, and Dingqi Yang. A survey on hypergraph representation learning. *ACM Computing Surveys*, 56(1):1–38, 2023.

[3] Naheed Anjum Arafat, Debabrota Basu, Laurent Decreusefond, and Stéphane Bressan. Construction and random generation of hypergraphs with prescribed degree and dimension sequences. In *International Conference on Database and Expert Systems Applications*, pages 130–145. Springer, 2020.

[4] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.

[5] Anna Badalyan, Nicolò Ruggeri, and Caterina De Bacco. Structure and inference in hypergraphs with node attributes. *Nature Communications*, 15(1):7073, 2024.

[6] Song Bai, Feihu Zhang, and Philip HS Torr. Hypergraph convolution and hypergraph attention. *Pattern Recognition*, 110:107637, 2021.

[7] Austin R. Benson, Rediet Abebe, Michael T. Schaub, Ali Jadbabaie, and Jon Kleinberg. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences*, 2018.

[8] Andreas Bergmeister, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. Efficient and scalable graph generation through iterative local expansion. In *12th International Conference on Learning Representations (ICLR), Vienna, Austria*, 2024.

[9] Hanqun Cao, Cheng Tan, Zhangyang Gao, Yilun Xu, Guangyong Chen, Pheng-Ann Heng, and Stan Z Li. A survey on generative diffusion models. *IEEE transactions on knowledge and data engineering*, 36(7):2814–2830, 2024.

[10] Ke Chen, Yusong Wu, Haohe Liu, Marianna Nezhurina, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. Musicldm: Enhancing novelty in text-to-music generation using beat-synchronous mixup strategies. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1206–1210. IEEE, 2024.

[11] Minyoung Choe, Jihoon Ko, Taehyung Kwon, Kijung Shin, and Christos Faloutsos. Kronecker generative models for power-law patterns in real-world hypergraphs. In *Proceedings of the ACM on Web Conference 2025*, pages 1261–1272, 2025.

[12] Jooyoung Choi, Jungbeom Lee, Chaehun Shin, Sungwon Kim, Hyunwoo Kim, and Sungroh Yoon. Perception prioritized training of diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11472–11481, 2022.

[13] Martina Contisciani, Federico Battiston, and Caterina De Bacco. Inference of hyperedges and overlapping communities in hypergraphs. *Nature communications*, 13(1):7229, 2022.

[14] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[15] Manh Tuan Do, Se-eun Yoon, Bryan Hooi, and Kijung Shin. Structural patterns and generative models of real-world hypergraphs. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 176–186, 2020.

[16] Dorian Gailhard, Enzo Tartaglione, Lirida Naviner, and Jhony H Giraldo. Hygene: A diffusion-based hypergraph generation method. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 16682–16690, 2025.

[17] Frédéric Giroire, Nicolas Nisse, Thibaud Trolliet, and Małgorzata Sulkowska. Preferential attachment hypergraph with high modularity. *Network Science*, 10(4):400–429, 2022.

[18] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[19] WK HASTINGS. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[20] Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2016.

[21] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[22] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in neural information processing systems*, 35:8633–8646, 2022.

[23] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

[24] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8107–8116, 2020.

[25] Sunwoo Kim, Fanchen Bu, Minyoung Choe, Jaemin Yoo, and Kijung Shin. How transitive are real-world group interactions?-measurement and reproduction. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1132–1143, 2023.

[26] Sunwoo Kim, Soo Yong Lee, Yue Gao, Alessia Antelmi, Mirko Polato, and Kijung Shin. A survey on hypergraph neural networks: An in-depth and step-by-step guide. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6534–6544. Association for Computing Machinery, 2024.

[27] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[28] Yunbum Kook, Jihoon Ko, and Kijung Shin. Evolution of real-world hypergraphs: Patterns and models without oracles. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 272–281. IEEE, 2020.

[29] Vladimir Kulikov, Shahar Yadin, Matan Kleiner, and Tomer Michaeli. Sinddm: A single image denoising diffusion model. In *International conference on machine learning*, pages 17920–17930. PMLR, 2023.

[30] Tarun Kumar, Sankaran Vaidyanathan, Harini Ananthapadmanabhan, Srinivasan Parthasarathy, and Balaraman Ravindran. Hypergraph clustering by iteratively reweighted modularity maximization. *Applied Network Science*, 5(1):52, 2020.

[31] Geon Lee, Fanchen Bu, Tina Eliassi-Rad, and Kijung Shin. A survey on hypergraph mining: Patterns, tools, and generators. *ACM Computing Surveys*, 57(8):1–36, 2025.

[32] Geon Lee, Minyoung Choe, and Kijung Shin. How do hyperedges overlap in real-world hypergraphs?-patterns, measures, and generators. In *Proceedings of the web conference 2021*, pages 3396–3407, 2021.

[33] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2–es, 2007.

[34] Stratis Limnios, Praveen Selvaraj, Mihai Cucuringu, Carsten Maple, Gesine Reinert, and Andrew Elliott. Sagess: A sampling graph denoising diffusion model for scalable graph generation. In *ECAI 2024*, pages 2950–2957. IOS Press, 2024.

[35] Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q Weinberger. Latent diffusion for language generation. *Advances in Neural Information Processing Systems*, 36:56998–57025, 2023.

[36] Rossana Mastrandrea, Julie Fournet, and Alain Barrat. Contact patterns in a high school: a comparison between data collected using wearable sensors, contact diaries and friendship surveys. *PloS one*, 10(9):e0136497, 2015.

[37] Kazuki Nakajima, Kazuyuki Shudo, and Naoki Masuda. Randomizing hypergraphs preserving degree correlation and local clustering. *IEEE Transactions on Network Science and Engineering*, 9(3):1139–1153, 2021.

[38] Mark EJ Newman. Assortative mixing in networks. *Physical review letters*, 89(20):208701, 2002.

[39] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021.

[40] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[41] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[42] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[43] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022.

[44] Juliette Stehlé, Nicolas Voirin, Alain Barrat, Ciro Cattuto, Lorenzo Isella, Jean-François Pinton, Marco Quaggiotto, Wouter Van den Broeck, Corinne Régis, Bruno Lina, et al. High-resolution measurements of face-to-face contact patterns in a primary school. *PloS one*, 6(8):e23176, 2011.

[45] Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *Journal of machine learning research*, 3(Dec):583–617, 2002.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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

[48] Weihuang Wen and Tianshu Yu. Hyperplr: Hypergraph generation through projection, learning, and reconstruction. In *The Thirteenth International Conference on Learning Representations*, 2025.

[49] Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 555–564, 2017.

[50] Cai Zhou, Xiyuan Wang, and Muhan Zhang. Unifying generation and prediction on graphs with latent graph diffusion. *Advances in Neural Information Processing Systems*, 37:61963–61999, 2024.

[51] Yanqiao Zhu, Yuanqi Du, Yinkai Wang, Yichen Xu, Jieyu Zhang, Qiang Liu, and Shu Wu. A survey on deep graph generation: Methods and applications. In *Learning on Graphs Conference*, pages 47–1. PMLR, 2022.

[52] Xianglin Zuo, Wenqi Chen, Xianduo Song, Xin Wang, and Ying Wang. Generating real-world hypergraphs via deep generative models. *Information Sciences*, 647:119412, 2023.

A Background and notation details

Hypergraphs generalize graphs by allowing each hyperedge to connect an arbitrary number of nodes, making them suitable for modeling higher-order interactions. Formally, a hypergraph is an ordered pair $\mathrm{H} = (\mathrm{V}, \mathrm{E})$, where V is the node set and E is a set of hyperedges, where $e \in \mathrm{E}$ and $e \subseteq \mathrm{V}$.

Incidence Matrix. A hypergraph $\mathrm{H} = (\mathrm{V}, \mathrm{E})$ can be represented by an incidence matrix $\mathbf{H} \in \{0, 1, \ldots, k-1\}^{|\mathrm{V}|\times|\mathrm{E}|}$, where $h_{ve} = 0$ indicates that node $v \in \mathrm{V}$ is not incident to hyperedge $e \in \mathrm{E}$, and $h_{ve} \in \{1, \ldots, k-1\}$ indicates incidence with class label h_{ve}. The case $k = 2$ recovers the standard binary incidence matrix.

Dual Hypergraph. Given $\mathrm{H} = (\mathrm{V}, \mathrm{E})$, its dual hypergraph is $\mathrm{H}^* = (\mathrm{V}^*, \mathrm{E}^*)$, where $\mathrm{V}^* := \mathrm{E}$ and $\mathrm{E}^* := \{e_v^* : v \in \mathrm{V}\}$ with $e_v^* := \{\, e \in \mathrm{E} : v \in e \,\} \subseteq \mathrm{V}^*$. Thus, the dual swaps the roles of nodes and hyperedges, and its incidence matrix satisfies $\mathbf{H}^* = \mathbf{H}^\top$.

Line Graph. The line graph of a hypergraph $\mathrm{H} = (\mathrm{V}, \mathrm{E})$ is the graph $\mathrm{LG}(\mathrm{H}) = (\mathrm{E}, \mathrm{E}_{\mathrm{LG}})$, where vertices correspond to hyperedges of H and two vertices are adjacent if the corresponding hyperedges intersect, i.e., $\{i, j\} \in \mathrm{E}_{\mathrm{LG}}$ if and only if $e_i \cap e_j \neq \emptyset$. Equivalently, it is the clique expansion of the dual hypergraph.

Node and Hyperedge Features. Let $\mathbf{X}^{\mathrm{V}} \in \mathbb{R}^{N\times d_{\mathrm{V}}}$ denote the node feature matrix, where the v-th row gives the features of node $v \in \mathrm{V}$, and let $\mathbf{X}^{\mathrm{E}} \in \mathbb{R}^{M\times d_{\mathrm{E}}}$ denote the hyperedge feature matrix, where the e-th row gives the features of hyperedge $e \in \mathrm{E}$. When features are unavailable on one side, they can be initialized as constant or learnable embeddings, or derived from the incidence structure.

Denoising Diffusion . Diffusion models are probabilistic generative models that approximate a data distribution $p(x)$ by learning how to reverse a gradual noising process. This corresponds to learning the reverse of a fixed Markov chain of length T:

$$q(x_{1:T} \mid x_0) = \prod_{t=1}^{T} q(x_t \mid x_{t-1}), \qquad q(x_t \mid x_{t-1}) = \mathcal{N}\big(x_t; \sqrt{1-\beta_t}\, x_{t-1},\, \beta_t \mathbf{I}\big),$$

with a variance schedule $\{\beta_t\}_{t=1}^T$. Defining $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t := \prod_{s=1}^t \alpha_s$, one can sample x_t in closed form as $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \varepsilon, \varepsilon \sim \mathcal{N}(0, I)$. The generative model learns the reverse transitions $p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\big(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)\big)$, so that sampling starts from $x_T \sim \mathcal{N}(0, I)$ and iteratively denoises to obtain x_0. This can be viewed as a hierarchy of denoising autoencoders sharing parameters across noise levels.

In practice, diffusion denoisers are most often trained to predict the injected noise ε [21]. However, the literature also reports improved training stability with v-*prediction* [43]. To avoid overloading the one-step noise coefficient $\alpha_t = 1 - \beta_t$, we define $a_t := \sqrt{\bar{\alpha}_t}, \qquad \sigma_t := \sqrt{1-\bar{\alpha}_t}$. The forward noising process can then be written as $x_t = a_t x_0 + \sigma_t \varepsilon$. The velocity target is $v_t := a_t \varepsilon - \sigma_t x_0$, and the model is trained with the objective $\mathbb{E}_{t,x_0,\varepsilon}\Big[\|v_t - \hat{v}_\theta(x_t, t)\|_2^2\Big]$. Given a prediction $\hat{v}_\theta$, the corresponding estimates of the clean sample and noise are

$$\hat{x}_{0,\theta} = a_t x_t - \sigma_t \hat{v}_\theta(x_t, t), \qquad \hat{\varepsilon}_\theta = \sigma_t x_t + a_t \hat{v}_\theta(x_t, t).$$

Latent Diffusion Models. Training diffusion models directly in the original data space can be expensive, since denoising must be repeated for many timesteps on high-dimensional signals. Latent Diffusion Models (LDMs) reduce this cost by moving the diffusion process to a learned representation space [41]. Concretely, an autoencoder (E, D) is trained such that $z = E(x)$ is a spatially downsampled (but information-preserving) representation and $D(z) \approx x$. A diffusion model is then trained on learned latents z. During generation, z_0 is sampled by the diffusion model and then decoded as $x = D(z_0)$. Beyond efficiency, the latent-space formulation also provides a convenient continuous interface when the original observations are non-continuous, as in the case of hypergraphs. Each object can instead be mapped to a continuous latent vector $z \in \mathbb{R}^d$, and standard continuous diffusion can then be applied in that space, relying on the decoder to map samples back to valid objects [35]. This strategy avoids discrete noising kernels and reverse-time categorical transitions, which are required by discrete denoising diffusion models [4].

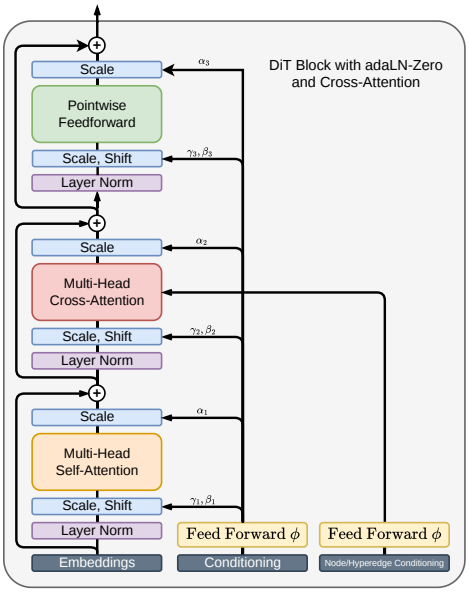


Figure 3: Overview of the Diffusion Transformer layer employed in the denoiser architecture of JANUS.

B Reproducibility and implementation details

Most diffusion models for image generation relied on U-Net backbones [42], leveraging convolutional inductive biases for local spatial structure. Diffusion Transformers (DiT) [40] replace the U-Net denoiser with a Transformer that operates on a sequence of latent patches, closely following Vision Transformer (ViT) design principles. In the standard DiT setup, a central empirical finding is that DiT quality improves predictably with scale.

In the following, we provide the full training protocol used in our experiments. Since each dataset is given as a single large hypergraph, we first construct a dataset of overlapping sub-hypergraphs as described in Sec. 4. In particular, we sample one sub-hypergraph for each hyperedge of the original hypergraph, yielding a collection of local training instances. We then reserve 10% of the sampled sub-hypergraphs for validation and use the remaining 90% for training. Unless otherwise stated, all mini-batches contain 32 sampled sub-hypergraphs.

To construct sampled sub-hypergraphs, we use the biased random walk procedure described in Sec. 4 on the line graph of the original hypergraph. In all experiments, the random walk uses parameters $p = 2.0$ and $q = 0.5$, producing walks of length fixed to 256. Empirically, this more explorative walk produced sub-hypergraphs that were easier to reconstruct during β-VAE training. Performing random walks over nodes of the line graph, which correspond to hyperedges, may naturally produce hypergraphs with variable numbers of nodes. We handle this by randomly sampling nodes that do not belong to the hypergraph and using them as padding.

For optimization, we do not use any learning-rate schedule or warmup. The β-VAE is trained for at least 3000 epochs. The InfoNCE temperature parameter τ is set to 1. The latent diffusion model is trained until convergence, which we determine by monitoring Hyper NetSimile and Hyperedge Portrait Divergence on generated validation samples and stopping once these metrics no longer improve. For the diffusion model, we additionally maintain an exponential moving average (EMA) of the denoiser parameters with decay 0.99. In all experiments, the diffusion process uses a *linear* noise schedule with $T = 1000$ denoising steps.

For each train/validation split, we compute node features on the training-only clique expansion and hyperedge features on the training-only line graph, after removing all validation hyperedges. These split-specific features are used for both training and validation samples. In both cases, we run `node2vec` on the input graph G with parameters $p = 2.0$, $q = 0.5$, 500 training epochs, and

batch size 1024. In preliminary experiments, we did not observe strong sensitivity of the downstream model to the exact `node2vec` hyperparameters.

Algorithm 1 summarizes the training procedure of the hypergraph β-VAE. Training alternates between incidence reconstruction in the latent space and node-consistency learning across overlapping sub-hypergraphs.

Algorithm 1: Training the Hypergraph β-VAE

Input: Sub-hypergraphs (X^V, X^E, H), KL weights β_V, β_E, flag UNCONSTRAINED
Output: Trained β-VAE encoder and decoder

```
Initialize the node encoder, hyperedge encoder, incidence decoder, and, if needed, node decoder
for each training epoch do
    for each mini-batch (X^V, X^E, H) do
        Encode nodes and hyperedges into latent variables Z^V and Z^E
        Reconstruct the incidence matrix from (Z^V, Z^E)
        Compute the incidence reconstruction loss and KL regularization losses
        if UNCONSTRAINED then
            Decode node representations for entity resolution
            Compute the node consistency loss across overlapping sub-hypergraphs
            if this is an incidence-reconstruction step then
                Backpropagate only the incidence reconstruction objective
            else
                Backpropagate only the node consistency objective
        else
            Backpropagate the incidence reconstruction objective
        Update the trainable parameters
    Evaluate the validation loss
Select the checkpoint with the best validation loss
return trained encoder and decoder
```

Algorithm 2 summarizes the training of the latent diffusion component. The pretrained β-VAE encoder provides clean node and hyperedge latents, and two coupled denoisers are trained with a standard v-prediction objective. In all experiments, we use a linear variance schedule with $T = 1000$ diffusion steps.

Algorithm 3 summarizes hypergraph generation at inference time. The model samples node and hyperedge latents by reverse diffusion and then decodes them into a valid incidence structure using the pretrained β-VAE decoder. This algorithm is applied at the sub-hypergraph level, and the resulting generated samples are merged into a global hypergraph using the entity-resolution procedure described in Sec. 4.

C Model hyperparameters

For the training of the β-VAE, we use the **S** configuration for all datasets except email-Eu. For email-Eu, which is the largest dataset in our benchmark in terms of both nodes and hyperedges, we use the larger **M** configuration. For the diffusion model, we use the **M** configuration on all datasets. The latent dimensionality is fixed to $d_z = 128$ in all experiments.

All experiments use a batch size of 32. The β-VAE is trained with $\beta = 10^{-6}$ following the same choice of [41], and learning rate 10^{-4}, without learning-rate scheduling or warmup. The β-VAE weight decay is set to 10^{-5}. The latent diffusion model is trained with learning rate 10^{-4}, a *linear* variance schedule, $T = 1000$ diffusion steps, weight decay 0, and EMA decay 0.99.

For feature extraction, we compute node features with `node2vec` on the clique expansion and hyperedge features with `node2vec` on the line graph. In both cases, `node2vec` is run with return parameter $p = 2.0$, in-out parameter $q = 0.5$, 500 epochs, and batch size 1024. In our experiments, the downstream model was not strongly sensitive to moderate changes in these `node2vec` hyperparameters.

Algorithm 2: Training the Hypergraph Latent Diffusion Model

Input: Trained β-VAE, sub-hypergraphs (X^V, X^E, H), diffusion steps T, flag UNCONSTRAINED
Output: Trained latent diffusion model

Freeze the pretrained β-VAE
Initialize the hyperedge denoiser
if UNCONSTRAINED **then**
    Initialize the node denoiser
for *each training epoch* **do**
    for *each mini-batch* (X^V, X^E, H) **do**
        Encode the mini-batch with the frozen β-VAE to obtain clean latents (Z^V, Z^E)
        Sample a diffusion step t
        Add Gaussian noise to the hyperedge latents Z^E to obtain Z_t^E
        if UNCONSTRAINED **then**
            Add Gaussian noise to the node latents Z^V to obtain Z_t^V
            Predict the node velocity target from Z_t^V conditioned on Z_t^E
            Predict the hyperedge velocity target from Z_t^E conditioned on Z_t^V
        else
            Predict the hyperedge velocity target from Z_t^E conditioned on the clean node latents Z^V
        Compute the denoising loss between predicted and true velocity targets
        Backpropagate the denoising loss
        Update the diffusion parameters with AdamW
        Update the EMA weights
Select the checkpoint with the best validation generation quality
return *trained latent diffusion model*

`node2vec` features in JANUSNC are computed on the clique expansion of the observed hypergraph. These features are therefore structural and transductive, and are used to match the projection-based node-set-constrained protocol adopted by HyperPLR. An inductive setting using only exogenous or topology-free node features is outside the scope of this evaluation.

For sub-hypergraph construction, the biased random walk sampler uses parameters $p = 2.0$ and $q = 0.5$. We found that this more explorative setting produced sampled sub-hypergraphs that were easier for the β-VAE to reconstruct during training.

C.1 Training and validation sub-hypergraphs

Because the training and validation sub-hypergraphs are sampled from the same observed hypergraph, some hyperedges may occur in samples from both splits. We therefore quantified this overlap for the 90% training and 10% validation splits we adopted in our experiments across all datasets. In Table 3 we report the mean pairwise Jaccard overlap to measure it.

Training and validation protocol. From each original hypergraph, we construct one sampled sub-hypergraph per hyperedge. We reserve 10% of the sampled sub-hypergraphs for validation and train on the remaining 90%. The sub-hypergraph sampler is a biased random walk with parameters $p = 2.0$ and $q = 0.5$. The β-VAE is trained for at least 3000 epochs, whereas the diffusion model is trained until the generated validation samples cease to improve according to Hyper NetSimile and Hyperedge Portrait Divergence. The diffusion process uses a linear noise schedule with $T = 1000$ steps. For feature extraction, `node2vec` is run with $p = 2.0$, $q = 0.5$, 500 epochs, and batch size 1024.

Algorithm 3: Sampling with the Hypergraph Latent Diffusion Model

Input: Trained latent diffusion model, pretrained β-VAE decoder, number of reverse steps T, flag UNCONSTRAINED
Output: Generated sub-hypergraph

if UNCONSTRAINED **then**
  Initialize node latents Z_T^V with Gaussian noise
else
  Take the input node features and map them to fixed node latents Z^V
Initialize hyperedge latents Z_T^E with Gaussian noise
for $t = T, T-1, \ldots, 1$ **do**
  if UNCONSTRAINED **then**
    Update the node latents by one reverse diffusion step, conditioned on the current hyperedge latents
  Update the hyperedge latents by one reverse diffusion step, conditioned on the current node latents
Decode the final latents (Z_0^V, Z_0^E) into incidence logits using the pretrained β-VAE decoder
Sample a discrete incidence matrix from the decoded logits
if UNCONSTRAINED **then**
  Decode node representations for entity resolution
Build the generated sub-hypergraph from the sampled incidence matrix
return *generated sub-hypergraph*

Table 1: β-VAE architecture and optimization hyperparameters. In the experiments, size S is used for all datasets except email-Eu, for which we use size M. The latent dimensionality is fixed to 128 in all cases.

Size	# Layers	# Heads	Hidden size	Latent dim.
S	1	2	128	128
M	2	4	256	128

Table 2: Latent diffusion model (DDM) architecture and optimization hyperparameters. In all experiments, we use size M. The latent dimensionality is fixed to 128.

Size	# Layers	# Heads	Hidden size	Latent dim.
S	4	2	256	128
M	8	4	512	128

D Datasets

We evaluate our method on five real-world hypergraph datasets selected from the collection used in this work. Together, these datasets span three application domains: communication, face-to-face proximity, and biomedical co-classification. The benchmark suite consists of email-Enron, email-Eu, contact-high-school, contact-primary-school, and NDC-classes. In all experiments, we treat each dataset as a static hypergraph.

Following the protocol described in Sec. 5, we compute node and hyperedge features using `node2vec` and retain only unique hyperedges. Table 4 reports the summary statistics of the resulting preprocessed hypergraphs used in our experiments. In particular, we report the number of nodes $|V|$, the number of unique hyperedges $|E|$, and the number of connected components in each dataset.

The communication portion of the benchmark comprises email-Enron and email-Eu. These two datasets provide email-based high-order interaction structure and allow us to evaluate generation quality on sparse communication hypergraphs of different scales. Among all datasets considered in this paper, email-Eu is the largest in terms of both nodes and hyperedges. After preprocessing, email-Enron contains 143 nodes and 1,512 unique hyperedges and forms a single connected component, whereas email-Eu contains 998 nodes and 25,027 unique hyperedges and is split into 20 connected components. The face-to-face proximity portion of the benchmark consists of contact-high-school

Table 3: Dataset statistics, train/validation splits, and average overlap.

Dataset	#V	#E	Train (90%)	Val (10%)	Avg. overlap (%)
NDC-classes	1,161	1,088	979	109	0.211898 ± 0.000005
contact-high-school	327	7,818	7,036	782	0.024797 ± 0.000075
contact-primary-school	242	12,704	11,433	1,271	0.015559 ± 0.000039
email-Enron	143	1,512	1,360	152	0.120826 ± 0.000573
email-Eu	998	25,027	22,524	2,503	0.097211 ± 0.000023

Table 4: Summary statistics of the hypergraphs datasets employed in our experiments.

Abbreviation	Dataset	$\lvert V\rvert$	$\lvert E\rvert$	# Connected Components
NDC	NDC-classes	1 161	1 088	183
CHS	contact-high-school	327	7 818	1
CPS	contact-primary-school	242	12 704	1
EE	email-Enron	143	1 512	1
EEu	email-Eu	998	25 027	20

and contact-primary-school. These datasets provide hypergraphs built from physical co-presence interactions and are useful for evaluating whether a model can capture recurring local group structure in social-contact data. After preprocessing, contact-high-school contains 327 nodes and 7,818 unique hyperedges, while contact-primary-school contains 242 nodes and 12,704 unique hyperedges. Both datasets are connected. The biomedical part of the benchmark is represented by NDC-classes. This dataset differs markedly from the communication and proximity datasets in both scale and connectivity structure. After preprocessing, it contains 1,161 nodes and 1,088 unique hyperedges distributed across 183 connected components. This makes NDC-classes the most fragmented dataset in our benchmark and provides a complementary test case for assessing generation quality on highly disconnected higher-order data. We selected this suite to cover heterogeneous higher-order systems with different sizes, densities, and connectivity patterns. In particular, the benchmark contains both fully connected hypergraphs (e.g., email-Enron, contact-high-school, and contact-primary-school) and highly disconnected ones (e.g., email-Eu and especially NDC-classes). This diversity is important for evaluating hypergraph generators across macroscopic, mesoscopic, and microscopic structural criteria. Each dataset is used as a real target hypergraph for unconditional and conditional generation experiments. For scalability, when needed, training is performed on overlapping sampled sub-hypergraphs derived from the original hypergraph, as described in Sec. 4.3. Evaluation, however, is always reported with respect to the corresponding real dataset summarized in Table 4.

E Evaluation framework implementation details

This appendix describes the evaluation taxonomy reported in Figure 4 used to compare real and generated hypergraphs. Let $\mathcal{H} = (V, E)$ be a hypergraph with node set V and hyperedge set $E \subseteq 2^V$. We denote by $n = |V|$ the number of nodes, by $m = |E|$ the number of hyperedges, and by $\mathbf{H} \in \{0,1\}^{n \times m}$ the binary incidence matrix, where $\mathbf{H}_{ve} = 1$ if and only if node v belongs to hyperedge e.

For each metric, we compute the value on the real hypergraph and on each generated hypergraph. For scalar metrics, the reported error is the absolute difference between the generated and real values. For distributional metrics, we compare the empirical distributions of the real and generated hypergraphs using the Wasserstein distance. We organize the evaluation metrics into three structural scales: microscopic, mesoscopic, and macroscopic; and two complementary evaluation categories: reconstruction and higher-order dissimilarity.

- **Microscopic metrics.** These metrics evaluate local structural properties of the generated hypergraph.
  - **Node degree distribution.** The degree of a node v is

$$d(v) = |\{e \in E : v \in e\}|.$$

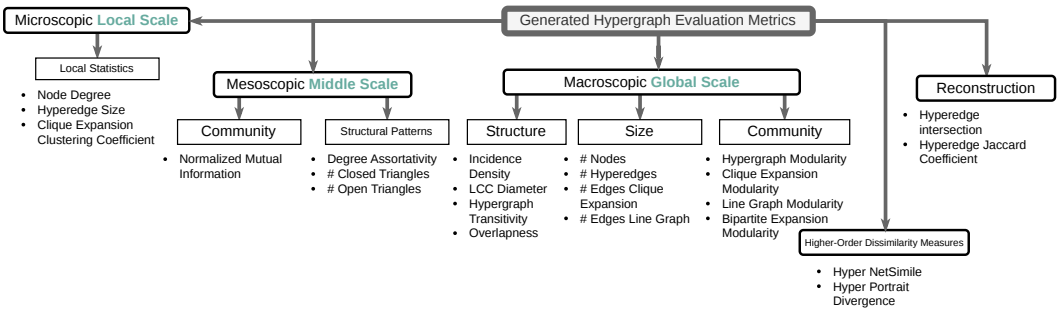


Figure 4: Figure 4: Overview of the evaluation metrics used in this work, organized into three structural scales (microscopic, mesoscopic, and macroscopic) and two complementary evaluation categories (reconstruction and higher-order dissimilarity).

We form the empirical distribution of node degrees by counting the frequency of each integer degree value. Real and generated distributions are compared using the Wasserstein distance.

  - **Hyperedge size distribution.** The size of a hyperedge e is its cardinality $|e|$. We form the empirical distribution of hyperedge sizes by counting the frequency of each integer cardinality value. Real and generated distributions are compared using the Wasserstein distance.
  - **Clique expansion clustering coefficient.** We first construct the clique expansion graph, where two nodes are adjacent if they appear together in at least one hyperedge. We then compute the average graph clustering coefficient on this projection, following the standard definition.

- **Mesoscopic metrics.** These metrics capture intermediate-scale organization, including communities and structural motifs.
  - **Normalized Mutual Information.** We detect communities directly on the hypergraph using the hypergraph modularity procedure [30] and compare the resulting node partitions using normalized mutual information (NMI) [45]. Higher values indicate stronger agreement between community assignments. We evaluate this metric only for the node-set constrained and reconstruction upper bound settings.
  - **Degree assortativity.** We measure degree assortative mixing in the hypergraph, following the degree assortativity coefficient for hypergraphs. This coefficient quantifies whether nodes with similar hypergraph degrees tend to participate in common high-order interactions, following the general assortativity principle of Newman [38].
  - **Closed triangles.** We count the number of triangles in the clique expansion graph. Equivalently, this is the number of unordered triples of nodes for which all three pairwise links appear in the projection. This statistic measures the extent to which high-order interactions induce closed three-node motifs.
  - **Open triangles.** We count unordered length-two paths in the clique expansion graph that do not close into triangles. This statistic complements closed triangles by measuring incomplete local triadic structures.
- **Macroscopic metrics.** These metrics describe global size, connectivity, density, and large-scale organization.
  - **Number of nodes.** We report $|V|$, the number of distinct nodes in the hypergraph. Real and generated values are compared by absolute difference.
  - **Number of hyperedges.** We report $|E|$, the number of distinct hyperedges after removing duplicates. Real and generated values are compared by absolute difference.
  - **Number of clique expansion edges.** We construct the clique expansion graph and count its edges. This measures how many distinct pairwise node relations are induced by the hyperedges.
  - **Number of line graph edges.** We construct the line graph, whose vertices are hyperedges and whose edges connect intersecting hyperedges. The number of line graph edges measures the amount of overlap among hyperedges.

- **Incidence density.** Given the incidence matrix $H \in \{0,1\}^{|V|\times|E|}$, incidence density is computed as
$$\rho(H) = \frac{|\{(v,e) : v \in e\}|}{|V||E|}.$$
It measures the fraction of nonzero entries in the incidence matrix.
- **Largest connected component diameter.** We extract the largest connected component of the hypergraph, project it through the clique expansion, and compute its graph diameter. This measures the maximum shortest-path distance within the largest connected part of the projected structure.
- **Largest connected component size.** We report both the number of nodes and the number of hyperedges in the largest connected component. These quantities summarize how much of the hypergraph is contained in its main connected region.
- **Hypergraph transitivity.** We use the hypergraph transitivity measure [25], which quantifies the tendency of group interactions to close higher-order relational patterns.
- **Overlapness.** We measure hyperedge overlapness [32]. This statistic captures how strongly hyperedges intersect beyond pairwise node co-occurrence. We report both average global overlapness and average egonet overlapness.
- **Clique expansion modularity.** We compute graph modularity on the clique expansion graph. Communities are obtained with Louvain graph partitioning, and modularity measures the strength of the resulting community structure among nodes under the pairwise projection.
- **Line graph modularity.** We compute graph modularity on the line graph. Communities are obtained with Louvain graph partitioning, and modularity measures community structure among hyperedges based on their intersections.
- **Bipartite expansion modularity.** We form the bipartite incidence graph with node set $V \cup E$ and edges (v,e) whenever $v \in e$. We then compute Louvain communities and graph modularity on this bipartite representation.
- **Hypergraph modularity.** We compute modularity directly on the hypergraph [30]. This evaluates community structure while preserving higher-order interactions instead of reducing the hypergraph to a pairwise graph.

- **Higher-order dissimilarity metrics.** These metrics compare real and generated hypergraphs through descriptors designed for higher-order networks.
  - **Hyper NetSimile.** Hyper NetSimile compares hypergraphs through feature vectors summarizing local and neighborhood-level higher-order properties. We compare the resulting feature vectors using Canberra distance, following the dissimilarity score [1]; lower values indicate greater similarity.
  - **Hyperedge Portrait Divergence.** Hyperedge Portrait Divergence compares hypergraphs using distributions of shortest-path and neighborhood structures over hyperedges. We compare the resulting hyperedge portraits using the squared Jensen–Shannon distance, following [1]; lower values indicate closer structural agreement.
- **Reconstruction metrics.** These metrics are used in the node-set constrained setting, where the generated and real hypergraphs are defined over the same node set.
  - **Hyperedge intersection.** We count the number of exactly recovered hyperedges:
$$|\hat{E} \cap E|.$$
Hyperedges are compared as unordered node sets. Higher values indicate better reconstruction of the observed hyperedge set.
  - **Hyperedge Jaccard coefficient.** We compute the Jaccard similarity between the generated and real hyperedge sets:
$$J(E,\hat{E}) = \frac{|E \cap \hat{E}|}{|E \cup \hat{E}|}.$$
As above, hyperedges are compared as unordered node sets. Higher values indicate stronger agreement between the two hyperedge sets.

F Detailed results

In this section, we report the full numerical results summarized in Sec. 5.1, covering micro-, meso-, and macro-scale metrics, higher-order dissimilarities, reconstruction scores, and the comparison with DGMH. We also report JANUS-BVAE and JANUSNC-BVAE, the β-VAE-only reconstruction variants of JANUS and JANUSNC. They omit latent diffusion and isolate the reconstruction capacity of the dual-view β-VAE: JANUS-BVAE reconstructs both nodes and hyperedges, while JANUSNC-BVAE reconstructs hyperedges on a fixed node set.

Table 5: Aggregated ranks over all the metric categories.

NDC-classes		**contact-high-school**		**contact-primary-school**		**email-Enron**		**email-Eu**	
Model	Grouped Rank	Model	Grouped Rank	Model	Grouped Rank	Model	Grouped Rank	Model	Grouped Rank
node set constrained		*node set constrained*		*node set constrained*		*node set constrained*		*node set constrained*	
JanusNC	1.17 ± 0.41	**JanusNC**	1.00 ± 0.00	**JanusNC**	1.00 ± 0.00	**JanusNC**	1.00 ± 0.00	**JanusNC**	1.50 ± 1.22
HyperPLR	2.50 ± 1.22	Hyperlap	3.17 ± 0.75	Hyperlap	3.17 ± 1.60	HyperPLR	2.67 ± 1.03	HyperDK11	3.00 ± 1.67
Hyperlap+	3.67 ± 0.52	HyperPLR	3.25 ± 1.25	HyperPLR	3.33 ± 1.51	HyperDK11	3.50 ± 1.22	HyperPLR	3.17 ± 1.47
HyperDK11	3.75 ± 1.84	Hyperlap+	3.67 ± 1.63	Hyperlap+	3.67 ± 0.82	Hyperlap+	4.17 ± 1.17	Hyperlap	3.67 ± 1.51
Hyperlap	4.75 ± 0.88	HyperDK11	4.08 ± 1.11	HyperDK11	4.00 ± 0.89	Hyperlap	4.33 ± 1.51	Hyperlap+	4.50 ± 0.84
HyperDK00	5.17 ± 1.33	HyperDK00	5.83 ± 0.41	HyperDK00	5.83 ± 0.41	HyperDK00	5.33 ± 1.03	HyperDK00	5.17 ± 1.33
unconstrained		*unconstrained*		*unconstrained*		*unconstrained*		*unconstrained*	
Janus	1.40 ± 0.55	**Janus**	1.30 ± 0.45	**Janus**	1.00 ± 0.00	**Janus**	1.00 ± 0.00	SubsetSampling	2.10 ± 0.74
TheRA	2.40 ± 1.14	TheRA	2.50 ± 1.22	HyperPA	2.40 ± 0.55	HyperPA	2.60 ± 0.89	**Janus**	2.30 ± 1.86
SubsetSampling	2.90 ± 1.14	HyperPA	2.90 ± 0.55	TheRA	2.80 ± 0.84	TheRA	3.00 ± 1.00	TheRA	3.10 ± 1.52
HyperPA	3.30 ± 0.84	SubsetSampling	3.30 ± 1.04	SubsetSampling	4.00 ± 0.71	SubsetSampling	3.80 ± 0.84	HyperPA	3.50 ± 1.12
NaivePA	5.00 ± 0.00	NaivePA	5.00 ± 0.00	NaivePA	4.80 ± 0.45	NaivePA	4.60 ± 0.89	NaivePA	4.00 ± 1.22

Table 5 summarizes the average ranking of each method across the evaluation categories used in the main paper. The table aggregates the detailed results reported below by grouping metrics into microscopic, mesoscopic, macroscopic, higher-order dissimilarity, and reconstruction categories. Methods are separated according to the experimental setting: unconstrained generation and node-set constrained generation.

Overall, the aggregated ranks show that the proposed diffusion-based models provide the most consistent performance across datasets and metric families. In the node-set constrained setting, JANUSNC obtains the best average rank on all five datasets. Its advantage is particularly clear on contact-high-school, contact-primary-school, and email-Enron, where it achieves rank 1.00 ± 0.00, indicating that it is ranked first in every aggregated category. On NDC-classes it remains almost uniformly best, with an average rank of 1.17 ± 0.41, while on email-Eu it is still the top method overall despite the larger variance (1.50 ± 1.22), reflecting the fact that this dataset is more challenging for some local statistics. The remaining constrained baselines occupy dataset-dependent positions: HYPERPLR is generally the strongest competitor, especially on NDC-classes and email-Enron, whereas HYPERDK11 is competitive on email-Eu. However, none of these methods matches the cross-category stability of JANUSNC.

A similar pattern appears in the unconstrained setting. JANUS achieves the best aggregated rank on four out of five datasets, with especially strong results on contact-primary-school and email-Enron, where it obtains rank 1.00 ± 0.00. On NDC-classes and contact-high-school, it also ranks first on average, although with slightly larger variance, suggesting occasional competition from structure-specific baselines. The only exception is email-Eu, where SUBSETSAMPLING obtains the best aggregated rank (2.10 ± 0.74), while JANUS ranks second (2.30 ± 1.86). This confirms the behavior observed in the detailed results: email-Eu is the hardest dataset for the proposed model, mainly because some baselines better preserve selected microscopic properties, but JANUS remains competitive when all evaluation categories are considered jointly.

The table also highlights the limitations of baselines that are designed around a single structural principle. THERA is the strongest unconstrained competitor on several datasets, consistent with its ability to reproduce transitivity-related structure, while HYPERPA and SUBSETSAMPLING can be competitive on specific local or preferential-attachment patterns. In the constrained setting, HYPERLAP, HYPERLAP+, HYPERDK, and HYPERPLR preserve selected overlap, local-degree, or reconstruction properties, but their ranks vary substantially across datasets and categories. In contrast, JANUS and JANUSNC achieve low average ranks with comparatively small standard deviations, showing that their advantage does not come from optimizing a single metric family, but from preserving hypergraph structure more uniformly across scales. This supports the main claim of the paper: jointly modeling node and hyperedge representations through a dual-view latent diffusion

Table 6: **Microscopic local statistics** performance comparison on the **NDC-classes** dataset. Methods are grouped by kind. Reported metrics: Clique Exp. Avg. Clustering (lower is better), Node Degree Dist. (lower is better), Hyperedge Size Dist. (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Clique Exp. Avg. Clustering	Node Degree Dist.	Hyperedge Size Dist.	Avg. Rank
unconstrained				
SubsetSampling	0.0888 ± 0.0096	4.7782 ± 0.8253	0.1899 ± 0.1359	1.6667
Janus	0.0460 ± 0.0109	5.0146 ± 0.6866	1.5250 ± 0.2165	2.6667
HyperPA	0.1331 ± 0.0131	6.8262 ± 0.7726	0.1471 ± 0.0414	3.0000
TheRA	0.1817	5.7460	0.3604	3.3333
NaivePA	0.2025 ± 0.0118	6.5787 ± 0.8933	0.9076 ± 0.1805	4.3333
(Janus) reconstruction upper bound				
JanusNC-β-VAE	0.0003 ± 0.0004	0.0215 ± 0.0149	0.0099 ± 0.0050	1.0000
Janus-β-VAE	0.0018 ± 0.0010	0.1602 ± 0.0184	0.0398 ± 0.0183	2.0000
node-set constrained				
HyperDK11	0.1414 ± 0.0040	0.1874 ± 0.0119	0.1587 ± 0.0049	2.0000
JanusNC	0.0485 ± 0.0046	1.3110 ± 0.0678	0.5228 ± 0.0487	2.3333
Hyperlap	0.0607 ± 0.0104	1.9195 ± 0.1087	0.2440 ± 0.0000	2.8333
Hyperlap+	0.1388 ± 0.0052	2.2368 ± 0.0595	0.2440 ± 0.0000	3.5000
HyperPLR	0.1796 ± 0.0098	1.3261 ± 0.1043	1.1018 ± 0.0358	4.3333
HyperDK00	0.3555 ± 0.0034	5.2728 ± 0.0411	2.0094 ± 0.0227	6.0000

Table 7: **Microscopic local statistics** performance comparison on the **contact-high-school** dataset. Methods are grouped by kind. Reported metrics: Clique Exp. Avg. Clustering (lower is better), Node Degree Dist. (lower is better), Hyperedge Size Dist. (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Clique Exp. Avg. Clustering	Node Degree Dist.	Hyperedge Size Dist.	Avg. Rank
unconstrained				
Janus	0.0315 ± 0.0048	2.9524 ± 0.4882	0.0690 ± 0.0023	2.0000
SubsetSampling	0.0128 ± 0.0120	27.4937 ± 1.9276	0.0168 ± 0.0044	2.0000
HyperPA	0.1057 ± 0.0056	31.4689 ± 1.5729	0.0037 ± 0.0019	2.6667
TheRA	0.1493	13.5076	0.2823	3.6667
NaivePA	0.1352 ± 0.0127	40.6730 ± 2.4702	0.9990 ± 0.0084	4.6667
(Janus) reconstruction upper bound				
JanusNC-β-VAE	0.0003 ± 0.0004	0.1058 ± 0.0195	0.0016 ± 0.0004	1.0000
Janus-β-VAE	0.0020 ± 0.0004	0.4606 ± 0.0672	0.0070 ± 0.0009	2.0000
node-set constrained				
JanusNC	0.0074 ± 0.0023	0.7314 ± 0.0568	0.0245 ± 0.0007	2.0000
Hyperlap+	0.1287 ± 0.0053	1.2159 ± 0.1165	0.0000 ± 0.0000	2.5000
Hyperlap	0.1625 ± 0.0039	1.0920 ± 0.0585	0.0000 ± 0.0000	2.8333
HyperPLR	0.0325 ± 0.0004	11.1189 ± 0.2333	0.4360 ± 0.0030	4.0000
HyperDK11	0.2099 ± 0.0020	2.5694 ± 0.0745	0.0156 ± 0.0007	4.3333
HyperDK00	0.1614 ± 0.0032	16.4820 ± 0.1588	0.7773 ± 0.0165	5.3333

process leads to more reliable hypergraph generation in both unconstrained and node-set constrained regimes.

F.1 Microscopic local statistics

Tables 6, 7, 8, 9, and 10 report microscopic local statistics for the NDC-classes, contact-high-school, contact-primary-school, email-Enron, and email-Eu datasets, respectively. These tables evaluate how well generated hypergraphs reproduce local structural properties of the real hypergraph. The reported

Table 8: **Microscopic local statistics** performance comparison on the **contact-primary-school** dataset. Methods are grouped by kind. Reported metrics: Clique Exp. Avg. Clustering (lower is better), Node Degree Dist. (lower is better), Hyperedge Size Dist. (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Clique Exp. Avg. Clustering	Node Degree Dist.	Hyperedge Size Dist.	Avg. Rank
unconstrained				
Janus	0.0084 ± 0.0020	5.5625 ± 0.1461	0.0311 ± 0.0032	1.6667
HyperPA	0.0261 ± 0.0131	67.0734 ± 0.6297	0.0059 ± 0.0056	2.0000
TheRA	0.0799	22.9917	0.3481	3.0000
SubsetSampling	0.1564 ± 0.0189	70.6643 ± 2.9423	0.0067 ± 0.0013	3.3333
NaivePA	0.2096 ± 0.0097	86.0562 ± 5.0824	0.9974 ± 0.0074	5.0000
(Janus) reconstruction upper bound				
JanusNC-β-VAE	0.0000 ± 0.0000	0.1810 ± 0.0486	0.0012 ± 0.0002	1.0000
Janus-β-VAE	0.0009 ± 0.0004	1.0091 ± 0.0713	0.0063 ± 0.0004	2.0000
node-set constrained				
JanusNC	0.0038 ± 0.0015	2.1124 ± 0.0861	0.0219 ± 0.0009	1.6667
Hyperlap	0.0292 ± 0.0009	2.2926 ± 0.3851	0.0000 ± 0.0000	1.8333
Hyperlap+	0.1290 ± 0.0229	4.3504 ± 1.1593	0.0000 ± 0.0000	3.1667
HyperDK11	0.0686 ± 0.0019	10.7215 ± 0.2688	0.0415 ± 0.0012	4.0000
HyperPLR	0.0345 ± 0.0000	16.0023 ± 0.4694	0.2867 ± 0.0041	4.3333
HyperDK00	0.1923 ± 0.0027	37.8463 ± 0.4755	0.7498 ± 0.0061	6.0000

Table 9: **Microscopic local statistics** performance comparison on the **email-Enron** dataset. Methods are grouped by kind. Reported metrics: Clique Exp. Avg. Clustering (lower is better), Node Degree Dist. (lower is better), Hyperedge Size Dist. (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Clique Exp. Avg. Clustering	Node Degree Dist.	Hyperedge Size Dist.	Avg. Rank
unconstrained				
Janus	0.0112 ± 0.0054	3.5287 ± 0.3674	0.3067 ± 0.0390	1.6667
HyperPA	0.0734 ± 0.0403	11.7011 ± 1.3787	0.0616 ± 0.0376	2.0000
TheRA	0.0805	10.6294	0.3776	3.0000
SubsetSampling	0.1391 ± 0.0114	13.8181 ± 1.5490	0.0883 ± 0.0253	3.6667
NaivePA	0.1589 ± 0.0112	12.8113 ± 2.5531	0.9956 ± 0.0580	4.6667
(Janus) reconstruction upper bound				
JanusNC-β-VAE	0.0004 ± 0.0009	0.0448 ± 0.0607	0.0024 ± 0.0033	1.0000
Janus-β-VAE	0.0014 ± 0.0016	0.0937 ± 0.0663	0.0035 ± 0.0027	2.0000
node-set constrained				
JanusNC	0.0021 ± 0.0018	0.8478 ± 0.1101	0.0493 ± 0.0064	1.3333
HyperDK11	0.0414 ± 0.0052	1.5804 ± 0.1058	0.0513 ± 0.0095	2.3333
Hyperlap	0.0524 ± 0.0087	2.5836 ± 0.3062	0.1217 ± 0.0000	3.5000
HyperPLR	0.0010 ± 0.0009	3.2313 ± 0.1835	0.4610 ± 0.0112	3.6667
Hyperlap+	0.2479 ± 0.0132	2.6126 ± 0.5885	0.1217 ± 0.0000	4.5000
HyperDK00	0.0868 ± 0.0029	14.4014 ± 0.2962	0.6538 ± 0.0255	5.6667

Table 10: **Microscopic local statistics** performance comparison on the **email-Eu** dataset. Methods are grouped by kind. Reported metrics: Clique Exp. Avg. Clustering (lower is better), Node Degree Dist. (lower is better), Hyperedge Size Dist. (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Clique Exp. Avg. Clustering	Node Degree Dist.	Hyperedge Size Dist.	Avg. Rank
unconstrained				
SubsetSampling	0.1597 ± 0.0130	30.5596 ± 5.3939	0.0275 ± 0.0073	1.6667
NaivePA	0.2368 ± 0.0048	49.8866 ± 2.2798	0.9942 ± 0.0263	2.6667
HyperPA	0.3406 ± 0.0116	81.1249 ± 0.6443	0.0181 ± 0.0052	3.3333
Janus	0.1319 ± 0.0023	64.0602 ± 0.7441	2.1892 ± 0.0087	3.3333
TheRA	0.3454	62.4391	1.2707	4.0000
(Janus) reconstruction upper bound				
JanusNC-β-VAE	0.0004 ± 0.0003	0.6631 ± 0.1053	0.0135 ± 0.0027	1.0000
Janus-β-VAE	0.0030 ± 0.0025	3.8457 ± 0.2356	0.0843 ± 0.0067	2.0000
node-set constrained				
HyperDK11	0.1710 ± 0.0047	2.9966 ± 0.1062	0.0803 ± 0.0025	2.0000
Hyperlap	0.1310 ± 0.0023	9.4033 ± 0.5780	0.1950 ± 0.0000	2.8333
HyperPLR	0.0455 ± 0.0025	6.4337 ± 0.5677	0.9769 ± 0.0014	3.0000
JanusNC	0.0286 ± 0.0059	23.5887 ± 0.5009	0.7265 ± 0.0072	3.3333
Hyperlap+	0.2936 ± 0.0044	9.9931 ± 0.4616	0.1950 ± 0.0000	4.1667
HyperDK00	0.2261 ± 0.0010	71.7068 ± 0.1027	0.9839 ± 0.0109	5.6667

Table 11: **Mesoscopic community** performance comparison on the **NDC-classes** dataset. Methods are grouped by kind. Reported metrics: Normalized Mutual Information (higher is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Normalized Mutual Information	Avg. Rank
(Janus) reconstruction upper bound		
JanusNC-β-VAE	0.8035 ± 0.0597	1.0000
Janus-β-VAE	0.5945 ± 0.0903	2.0000
node-set constrained		
JanusNC	0.4520 ± 0.0104	1.0000
HyperPLR	0.3794 ± 0.0107	2.0000
HyperDK00	0.2567 ± 0.0017	3.0000
HyperDK11	0.2466 ± 0.0094	4.0000
Hyperlap+	0.2382 ± 0.0014	5.0000
Hyperlap	0.1882 ± 0.0055	6.0000

metrics are the clique-expansion average clustering coefficient, the node degree distribution distance, and the hyperedge size distribution distance. Lower values indicate better agreement with the real dataset, and the average rank summarizes each method's performance across the local metrics.

F.2 Mesoscopic community

Tables 11, 12, 13, 14, and 15 report mesoscopic community-recovery results for the NDC-classes, contact-high-school, contact-primary-school, email-Enron, and email-Eu datasets, respectively. These tables measure whether the generated hypergraphs preserve the community organization of the original data. The reported metric is normalized mutual information, which compares the community partition induced by each generated hypergraph with that of the real hypergraph. Higher values indicate better community agreement, while the average rank summarizes the relative performance of the methods within each setting.

Table 12: **Mesoscopic community** performance comparison on the **contact-high-school** dataset. Methods are grouped by kind. Reported metrics: Normalized Mutual Information (higher is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Normalized Mutual Information	Avg. Rank
(Janus) reconstruction upper bound		
JanusNC-β-VAE	0.8561 ± 0.0000	1.0000
Janus-β-VAE	0.6284 ± 0.3043	2.0000
node-set constrained		
JanusNC	0.1266 ± 0.0352	1.0000
HyperDK00	0.0414 ± 0.0053	2.0000
HyperDK11	0.0323 ± 0.0110	3.0000
HyperPLR	0.0289 ± 0.0055	4.0000
Hyperlap	0.0276 ± 0.0080	5.0000
Hyperlap+	0.0219 ± 0.0039	6.0000

Table 13: **Mesoscopic community** performance comparison on the **contact-primary-school** dataset. Methods are grouped by kind. Reported metrics: Normalized Mutual Information (higher is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Normalized Mutual Information	Avg. Rank
(Janus) reconstruction upper bound		
JanusNC-β-VAE	0.6708 ± 0.1868	1.0000
Janus-β-VAE	0.3403 ± 0.3302	2.0000
node-set constrained		
JanusNC	0.1207 ± 0.0400	1.0000
HyperDK00	0.0608 ± 0.0231	2.0000
HyperDK11	0.0404 ± 0.0078	3.0000
HyperPLR	0.0365 ± 0.0048	4.0000
Hyperlap+	0.0248 ± 0.0095	5.0000
Hyperlap	0.0233 ± 0.0062	6.0000

Table 14: **Mesoscopic community** performance comparison on the **email-Enron** dataset. Methods are grouped by kind. Reported metrics: Normalized Mutual Information (higher is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Normalized Mutual Information	Avg. Rank
(Janus) reconstruction upper bound		
JanusNC-β-VAE	0.7338 ± 0.0527	1.0000
Janus-β-VAE	0.7131 ± 0.1132	2.0000
node-set constrained		
JanusNC	0.1417 ± 0.0384	1.0000
HyperDK11	0.0839 ± 0.0241	2.0000
HyperPLR	0.0821 ± 0.0160	3.0000
HyperDK00	0.0723 ± 0.0209	4.0000
Hyperlap+	0.0599 ± 0.0206	5.0000
Hyperlap	0.0564 ± 0.0167	6.0000

Table 15: **Mesoscopic community** performance comparison on the **email-Eu** dataset. Methods are grouped by kind. Reported metrics: Normalized Mutual Information (higher is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Normalized Mutual Information	Avg. Rank
(Janus) reconstruction upper bound		
JanusNC-β-VAE	0.4873 ± 0.1893	1.0000
Janus-β-VAE	0.0754 ± 0.0317	2.0000
node-set constrained		
JanusNC	0.0562 ± 0.0064	1.0000
HyperDK11	0.0378 ± 0.0033	2.0000
HyperPLR	0.0371 ± 0.0012	3.0000
HyperDK00	0.0320 ± 0.0017	4.0000
Hyperlap+	0.0291 ± 0.0031	5.0000
Hyperlap	0.0272 ± 0.0022	6.0000

Table 16: **Mesoscopic structural patterns** performance comparison on the **NDC-classes** dataset. Methods are grouped by kind. Reported metrics: Hypergraph Degree Assortativity (lower is better), # Closed Triangles (lower is better), # Open Triangles (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Hypergraph Degree Assortativity	# Closed Triangles	# Open Triangles	Avg. Rank
unconstrained				
Janus	0.1666 ± 0.0454	1638.8000 ± 1184.8674	1229.2000 ± 726.4384	1.3333
TheRA	0.1268	63676.0000	2717.0000	2.6667
HyperPA	0.5059 ± 0.0100	9521.2000 ± 2744.1732	4590.0000 ± 864.6465	3.3333
SubsetSampling	0.4480 ± 0.0130	42097.0000 ± 5611.2747	45282.2000 ± 7923.7922	3.3333
NaivePA	0.4508 ± 0.0101	58365.2000 ± 9078.8307	$83767.2000 \pm 16902.7303$	4.3333
(Janus) reconstruction upper bound				
JanusNC-β-VAE	0.0669 ± 0.0171	19.2000 ± 31.8700	6.4000 ± 9.7365	1.3333
Janus-β-VAE	0.0392 ± 0.0305	349.4000 ± 136.0121	60.2000 ± 76.0375	1.6667
node-set constrained				
JanusNC	0.0450 ± 0.0168	69.4000 ± 53.8359	11.6000 ± 14.2934	1.0000
HyperPLR	0.2684 ± 0.0040	4295.4000 ± 58.1016	7115.6000 ± 7.9875	2.3333
HyperDK00	0.4572 ± 0.0082	6289.2000 ± 471.9817	3508.8000 ± 280.3457	3.0000
Hyperlap	0.4122 ± 0.0150	81836.4000 ± 1011.5262	61507.8000 ± 3405.4202	4.6667
Hyperlap+	0.4580 ± 0.0456	66595.0000 ± 1550.1950	19531.6000 ± 1953.4395	4.6667
HyperDK11	0.4794 ± 0.0034	60097.6000 ± 1117.6052	88940.6000 ± 1430.5832	5.3333

F.3 Mesoscopic structural patterns

Tables 16, 17, 18, 19, and 20 report mesoscopic structural-pattern results for the NDC-classes, contact-high-school, contact-primary-school, email-Enron, and email-Eu datasets, respectively. These tables assess whether generated hypergraphs reproduce middle-scale structural organization beyond local degree and size statistics. The reported metrics are hypergraph degree assortativity, the number of closed triangles, and the number of open triangles, all expressed as errors with respect to the real hypergraph. Lower values indicate closer agreement with the original dataset, and the average rank summarizes performance across the structural-pattern metrics.

F.4 Macroscopic community

Tables 21, 22, 23, 24, and 25 report macroscopic community-structure results for the NDC-classes, contact-high-school, contact-primary-school, email-Enron, and email-Eu datasets, respectively. These tables compare the global modular organization of real and generated hypergraphs under complementary views of the same data. The reported metrics are clique-expansion modularity, hypergraph modularity, bipartite modularity, and line-graph modularity, reported as absolute errors from the real

Table 17: **Mesoscopic structural patterns** performance comparison on the **contact-high-school** dataset. Methods are grouped by kind. Reported metrics: Hypergraph Degree Assortativity (lower is better), # Closed Triangles (lower is better), # Open Triangles (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Hypergraph Degree Assortativity	# Closed Triangles	# Open Triangles	Avg. Rank
unconstrained				
Janus	0.0301 ± 0.0159	126.8000 ± 12.4780	4793.2000 ± 194.9172	1.3333
HyperPA	0.6444 ± 0.0363	269.4000 ± 115.0187	4506.4000 ± 3070.5346	2.6667
SubsetSampling	0.1427 ± 0.0241	4175.6000 ± 155.7524	19932.4000 ± 3405.5564	3.3333
TheRA	0.4547	2664.0000	18683.0000	3.3333
NaivePA	0.1781 ± 0.0154	11815.0000 ± 653.4635	$190303.0000 \pm 13624.2036$	4.3333
(Janus) reconstruction upper bound				
JanusNC-β-VAE	0.0199 ± 0.0237	9.4000 ± 3.7815	8.6000 ± 8.9051	1.3333
Janus-β-VAE	0.0176 ± 0.0121	72.8000 ± 13.3492	52.6000 ± 27.9517	1.6667
node-set constrained				
JanusNC	0.0186 ± 0.0109	7.4000 ± 4.2778	1744.2000 ± 107.8643	1.0000
HyperPLR	0.1022 ± 0.0271	3122.6000 ± 23.7023	13919.8000 ± 78.8365	3.0000
HyperDK11	0.1544 ± 0.0106	638.4000 ± 4.2190	47846.2000 ± 737.0853	3.3333
Hyperlap	0.1571 ± 0.0108	644.4000 ± 3.9115	19855.2000 ± 233.4603	4.0000
Hyperlap+	0.2067 ± 0.0355	640.4000 ± 5.6391	31592.6000 ± 1138.7477	4.3333
HyperDK00	0.1561 ± 0.0047	13882.2000 ± 432.7519	$164168.4000 \pm 4677.3007$	5.3333

Table 18: **Mesoscopic structural patterns** performance comparison on the **contact-primary-school** dataset. Methods are grouped by kind. Reported metrics: Hypergraph Degree Assortativity (lower is better), # Closed Triangles (lower is better), # Open Triangles (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Hypergraph Degree Assortativity	# Closed Triangles	# Open Triangles	Avg. Rank
unconstrained				
Janus	0.0801 ± 0.0480	63.6000 ± 46.4898	7019.4000 ± 121.6955	1.0000
HyperPA	0.4062 ± 0.0626	392.4000 ± 162.4448	10614.8000 ± 6508.8942	3.0000
SubsetSampling	0.1410 ± 0.0288	3479.6000 ± 402.7975	$17587.4000 \pm 13833.6117$	3.0000
TheRA	0.3084	3188.0000	23776.0000	3.6667
NaivePA	0.1654 ± 0.0128	17966.4000 ± 1516.3582	$234592.8000 \pm 35341.2282$	4.3333
(Janus) reconstruction upper bound				
JanusNC-β-VAE	0.0397 ± 0.0297	16.2000 ± 3.6332	34.6000 ± 36.6920	1.3333
Janus-β-VAE	0.0196 ± 0.0190	110.6000 ± 5.8566	472.6000 ± 139.8689	1.6667
node-set constrained				
JanusNC	0.0589 ± 0.0234	29.8000 ± 10.1833	3024.2000 ± 568.3350	1.3333
HyperPLR	0.0132 ± 0.0103	3094.8000 ± 52.2752	39809.8000 ± 52.2752	3.0000
Hyperlap+	0.1667 ± 0.0083	159.0000 ± 49.4823	$29547.6000 \pm 15052.6589$	3.3333
Hyperlap	0.1570 ± 0.0102	653.8000 ± 14.1315	41781.6000 ± 1997.6253	4.0000
HyperDK11	0.1560 ± 0.0060	835.2000 ± 6.3008	$270809.0000 \pm 2079.5510$	4.3333
HyperDK00	0.1463 ± 0.0055	23663.8000 ± 649.0498	$729303.0000 \pm 9079.2175$	5.0000

Table 19: **Mesoscopic structural patterns** performance comparison on the **email-Enron** dataset. Methods are grouped by kind. Reported metrics: Hypergraph Degree Assortativity (lower is better), # Closed Triangles (lower is better), # Open Triangles (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Hypergraph Degree Assortativity	# Closed Triangles	# Open Triangles	Avg. Rank
unconstrained				
Janus	0.0225 ± 0.0173	115.2000 ± 111.5267	52.8000 ± 32.5223	1.0000
HyperPA	0.3035 ± 0.0161	2788.2000 ± 642.6027	3021.6000 ± 1355.3648	3.0000
SubsetSampling	0.1923 ± 0.0310	4195.6000 ± 947.8480	6291.6000 ± 1888.3376	3.0000
TheRA	0.0672	15588.0000	8665.0000	3.6667
NaivePA	0.2913 ± 0.0099	4252.0000 ± 998.4415	21891.2000 ± 3146.9657	4.3333
(Janus) reconstruction upper bound				
JanusNC-β-VAE	0.0368 ± 0.0115	9.0000 ± 20.1246	4.8000 ± 10.7331	1.3333
Janus-β-VAE	0.0562 ± 0.0316	2.4000 ± 2.3022	6.8000 ± 8.1056	1.6667
node-set constrained				
JanusNC	0.0367 ± 0.0202	4.8000 ± 4.7645	81.8000 ± 24.8938	1.0000
HyperPLR	0.1523 ± 0.0124	1465.0000 ± 37.7624	2011.2000 ± 58.4953	2.6667
Hyperlap+	0.2685 ± 0.0170	13682.8000 ± 758.4070	1323.6000 ± 308.0321	3.6667
HyperDK00	0.2760 ± 0.0156	208.4000 ± 114.8011	46651.0000 ± 838.4211	4.3333
Hyperlap	0.2677 ± 0.0115	16949.6000 ± 536.6850	15849.4000 ± 2582.9575	4.3333
HyperDK11	0.2845 ± 0.0091	4066.2000 ± 160.9323	37831.4000 ± 974.1536	5.0000

Table 20: **Mesoscopic structural patterns** performance comparison on the **email-Eu** dataset. Methods are grouped by kind. Reported metrics: Hypergraph Degree Assortativity (lower is better), # Closed Triangles (lower is better), # Open Triangles (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Hypergraph Degree Assortativity	# Closed Triangles	# Open Triangles	Avg. Rank
unconstrained				
HyperPA	0.2943 ± 0.0130	68808.6000 ± 3983.7970	$89072.8000 \pm 13519.0941$	2.3333
TheRA	0.0184	727153.0000	745023.0000	2.3333
SubsetSampling	0.1863 ± 0.0163	$252536.0000 \pm 14546.0439$	$1460210.8000 \pm 200863.4878$	2.6667
NaivePA	0.1972 ± 0.0088	$314780.2000 \pm 13796.5497$	$2009051.8000 \pm 73805.9843$	3.6667
Janus	0.0377 ± 0.0121	$792608.6667 \pm 27584.7766$	$2876772.6667 \pm 41937.5387$	4.0000
(Janus) reconstruction upper bound				
JanusNC-β-VAE	0.0713 ± 0.0404	840.4000 ± 251.3708	1087.8000 ± 417.5598	1.3333
Janus-β-VAE	0.0649 ± 0.0464	17319.6000 ± 1658.2421	$100531.6000 \pm 8921.3966$	1.6667
node-set constrained				
JanusNC	0.1381 ± 0.0422	2448.6000 ± 1077.1770	59049.4000 ± 1885.1407	1.0000
HyperPLR	0.2044 ± 0.0086	127367.4000 ± 155.7861	$165975.6000 \pm 1367.2133$	3.0000
Hyperlap+	0.1762 ± 0.0355	$504741.4000 \pm 11561.8625$	$673393.8000 \pm 20347.2001$	3.3333
HyperDK00	0.2067 ± 0.0023	5562.0000 ± 1384.6366	$2467127.0000 \pm 29901.9227$	3.6667
Hyperlap	0.1974 ± 0.0065	$718826.6000 \pm 15624.4865$	$2837196.4000 \pm 115211.6587$	4.6667
HyperDK11	0.2148 ± 0.0007	$417648.4000 \pm 4489.0074$	$5279779.4000 \pm 22907.1066$	5.3333

Table 21: **Macroscopic community** performance comparison on the **NDC-classes** dataset. Methods are grouped by kind. Reported metrics: Clique Exp. Modularity (lower is better), Hypergraph Modularity (lower is better), Bipartite Modularity (lower is better), Line Graph Modularity (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Clique Exp. Modularity	Hypergraph Modularity	Bipartite Modularity	Line Graph Modularity	Avg. Rank
unconstrained					
Janus	0.0715 ± 0.0122	0.1282 ± 0.0188	0.1147 ± 0.0175	0.1099 ± 0.0210	1.0000
TheRA	0.0938	0.1881	0.1516	0.1519	2.0000
SubsetSampling	0.4472 ± 0.0123	0.4062 ± 0.0297	0.3553 ± 0.0232	0.3567 ± 0.0236	3.0000
HyperPA	0.4777 ± 0.0098	0.5045 ± 0.0212	0.4701 ± 0.0126	0.4673 ± 0.0115	4.2500
NaivePA	0.5047 ± 0.0070	0.5370 ± 0.0173	0.4691 ± 0.0123	0.4685 ± 0.0139	4.7500
(Janus) reconstruction upper bound					
JanusNC-β-VAE	0.0031 ± 0.0026	0.0059 ± 0.0028	0.0028 ± 0.0016	0.0016 ± 0.0011	1.2500
Janus-β-VAE	0.0064 ± 0.0062	0.0060 ± 0.0027	0.0017 ± 0.0009	0.0019 ± 0.0009	1.7500
node-set constrained					
HyperPLR	0.0128 ± 0.0068	0.0198 ± 0.0042	0.0121 ± 0.0102	0.0072 ± 0.0032	1.5000
JanusNC	0.0119 ± 0.0035	0.0111 ± 0.0041	0.0154 ± 0.0024	0.0138 ± 0.0013	1.5000
Hyperlap+	0.1821 ± 0.0488	0.1187 ± 0.0279	0.0943 ± 0.0332	0.0920 ± 0.0366	3.2500
Hyperlap	0.1571 ± 0.0195	0.1497 ± 0.0139	0.1632 ± 0.0171	0.1581 ± 0.0164	3.7500
HyperDK00	0.3818 ± 0.0050	0.3813 ± 0.0052	0.3394 ± 0.0025	0.3408 ± 0.0025	5.0000
HyperDK11	0.4252 ± 0.0057	0.4151 ± 0.0084	0.3439 ± 0.0018	0.3440 ± 0.0016	6.0000

Table 22: **Macroscopic community** performance comparison on the **contact-high-school** dataset. Methods are grouped by kind. Reported metrics: Clique Exp. Modularity (lower is better), Hypergraph Modularity (lower is better), Bipartite Modularity (lower is better), Line Graph Modularity (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Clique Exp. Modularity	Hypergraph Modularity	Bipartite Modularity	Line Graph Modularity	Avg. Rank
unconstrained					
Janus	0.0109 ± 0.0058	0.0450 ± 0.0034	0.0116 ± 0.0056	0.0044 ± 0.0019	1.0000
TheRA	0.1980	0.0776	0.0784	0.0783	2.0000
HyperPA	0.4910 ± 0.0023	0.5052 ± 0.0126	0.2841 ± 0.0017	0.2838 ± 0.0017	3.2500
SubsetSampling	0.4185 ± 0.0048	0.5078 ± 0.0042	0.2906 ± 0.0030	0.2905 ± 0.0030	3.7500
NaivePA	0.5214 ± 0.0020	0.5904 ± 0.0304	0.4264 ± 0.0012	0.4261 ± 0.0018	5.0000
(Janus) reconstruction upper bound					
JanusNC-β-VAE	0.0009 ± 0.0002	0.0034 ± 0.0044	0.0020 ± 0.0016	0.0016 ± 0.0011	1.2500
Janus-β-VAE	0.0047 ± 0.0064	0.0029 ± 0.0037	0.0026 ± 0.0024	0.0027 ± 0.0013	1.7500
node-set constrained					
JanusNC	0.0011 ± 0.0007	0.0189 ± 0.0038	0.0005 ± 0.0004	0.0040 ± 0.0013	1.2500
Hyperlap+	0.1608 ± 0.0016	0.0776 ± 0.0012	0.0052 ± 0.0009	0.0037 ± 0.0019	2.0000
HyperPLR	0.0271 ± 0.0016	0.1652 ± 0.0021	0.1009 ± 0.0055	0.1014 ± 0.0032	3.0000
Hyperlap	0.1924 ± 0.0134	0.1544 ± 0.0049	0.1304 ± 0.0031	0.1305 ± 0.0029	3.7500
HyperDK11	0.4983 ± 0.0034	0.5135 ± 0.0021	0.2950 ± 0.0002	0.2948 ± 0.0005	5.2500
HyperDK00	0.5154 ± 0.0014	0.5110 ± 0.0095	0.3778 ± 0.0007	0.3773 ± 0.0017	5.7500

Table 23: **Macroscopic community** performance comparison on the **contact-primary-school** dataset. Methods are grouped by kind. Reported metrics: Clique Exp. Modularity (lower is better), Hypergraph Modularity (lower is better), Bipartite Modularity (lower is better), Line Graph Modularity (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Clique Exp. Modularity	Hypergraph Modularity	Bipartite Modularity	Line Graph Modularity	Avg. Rank
unconstrained					
Janus	0.0119 ± 0.0025	0.0224 ± 0.0018	0.0103 ± 0.0016	0.0109 ± 0.0024	1.0000
TheRA	0.1358	0.1713	0.1219	0.1224	2.0000
HyperPA	0.2179 ± 0.0057	0.3517 ± 0.0297	0.2008 ± 0.0032	0.2004 ± 0.0021	3.2500
SubsetSampling	0.1531 ± 0.0101	0.3656 ± 0.0126	0.2102 ± 0.0020	0.2101 ± 0.0013	3.7500
NaivePA	0.2409 ± 0.0020	0.3951 ± 0.0142	0.3309 ± 0.0009	0.3308 ± 0.0010	5.0000
(Janus) reconstruction upper bound					
Janus-β-VAE	0.0010 ± 0.0003	0.0002 ± 0.0001	0.0007 ± 0.0006	0.0016 ± 0.0012	1.5000
JanusNC-β-VAE	0.0001 ± 0.0001	0.0002 ± 0.0002	0.0010 ± 0.0009	0.0008 ± 0.0006	1.5000
node-set constrained					
JanusNC	0.0058 ± 0.0030	0.0123 ± 0.0009	0.0023 ± 0.0007	0.0031 ± 0.0030	1.5000
Hyperlap	0.0034 ± 0.0030	0.0549 ± 0.0029	0.0156 ± 0.0019	0.0165 ± 0.0019	2.0000
HyperPLR	0.0026 ± 0.0009	0.0603 ± 0.0031	0.0352 ± 0.0018	0.0352 ± 0.0015	3.0000
Hyperlap+	0.2123 ± 0.0006	0.0866 ± 0.0083	0.0207 ± 0.0141	0.0211 ± 0.0142	3.5000
HyperDK11	0.2402 ± 0.0012	0.3610 ± 0.0127	0.2159 ± 0.0003	0.2155 ± 0.0003	5.0000
HyperDK00	0.2538 ± 0.0010	0.3686 ± 0.0256	0.2961 ± 0.0008	0.2958 ± 0.0005	6.0000

Table 24: **Macroscopic community** performance comparison on the **email-Enron** dataset. Methods are grouped by kind. Reported metrics: Clique Exp. Modularity (lower is better), Hypergraph Modularity (lower is better), Bipartite Modularity (lower is better), Line Graph Modularity (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Clique Exp. Modularity	Hypergraph Modularity	Bipartite Modularity	Line Graph Modularity	Avg. Rank
unconstrained					
Janus	0.0087 ± 0.0027	0.0047 ± 0.0039	0.0214 ± 0.0076	0.0235 ± 0.0106	1.0000
TheRA	0.1929	0.1377	0.0779	0.0734	2.2500
HyperPA	0.2297 ± 0.0136	0.3770 ± 0.0149	0.2626 ± 0.0138	0.2575 ± 0.0155	3.2500
SubsetSampling	0.0637 ± 0.0303	0.3964 ± 0.0411	0.2690 ± 0.0065	0.2668 ± 0.0060	3.5000
NaivePA	0.2737 ± 0.0051	0.4661 ± 0.0343	0.3612 ± 0.0059	0.3596 ± 0.0063	5.0000
(Janus) reconstruction upper bound					
Janus-β-VAE	0.0096 ± 0.0072	0.0006 ± 0.0003	0.0014 ± 0.0009	0.0030 ± 0.0033	1.5000
JanusNC-β-VAE	0.0068 ± 0.0036	0.0003 ± 0.0002	0.0050 ± 0.0046	0.0068 ± 0.0032	1.5000
node-set constrained					
JanusNC	0.0067 ± 0.0035	0.0042 ± 0.0018	0.0060 ± 0.0025	0.0069 ± 0.0045	1.2500
Hyperlap	0.0693 ± 0.0459	0.0246 ± 0.0065	0.0808 ± 0.0126	0.0775 ± 0.0117	2.2500
HyperPLR	0.0060 ± 0.0035	0.0794 ± 0.0109	0.0901 ± 0.0035	0.0852 ± 0.0041	3.0000
Hyperlap+	0.2379 ± 0.0085	0.2047 ± 0.0010	0.0817 ± 0.0013	0.0846 ± 0.0013	3.5000
HyperDK11	0.2666 ± 0.0025	0.3731 ± 0.0044	0.2771 ± 0.0010	0.2755 ± 0.0021	5.0000
HyperDK00	0.2724 ± 0.0027	0.4176 ± 0.0474	0.2969 ± 0.0021	0.2950 ± 0.0021	6.0000

Table 25: **Macroscopic community** performance comparison on the **email-Eu** dataset. Methods are grouped by kind. Reported metrics: Clique Exp. Modularity (lower is better), Hypergraph Modularity (lower is better), Bipartite Modularity (lower is better), Line Graph Modularity (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Clique Exp. Modularity	Hypergraph Modularity	Bipartite Modularity	Line Graph Modularity	Avg. Rank
unconstrained					
Janus	0.1156 ± 0.0076	0.0174 ± 0.0044	0.1060 ± 0.0017	0.1067 ± 0.0020	1.0000
TheRA	0.2714	0.2836	0.1133	0.1169	2.0000
SubsetSampling	0.3097 ± 0.0031	0.4463 ± 0.0258	0.3493 ± 0.0034	0.3507 ± 0.0035	3.2500
HyperPA	0.2992 ± 0.0036	0.5012 ± 0.0315	0.3629 ± 0.0013	0.3641 ± 0.0014	3.7500
NaivePA	0.3132 ± 0.0051	0.5091 ± 0.0077	0.4267 ± 0.0017	0.4275 ± 0.0018	5.0000
(Janus) reconstruction upper bound					
JanusNC-β-VAE	0.0019 ± 0.0007	0.0017 ± 0.0014	0.0031 ± 0.0014	0.0022 ± 0.0007	1.2500
Janus-β-VAE	0.0089 ± 0.0046	0.0012 ± 0.0008	0.0135 ± 0.0077	0.0033 ± 0.0024	1.7500
node-set constrained					
JanusNC	0.0193 ± 0.0042	0.0667 ± 0.0048	0.0397 ± 0.0029	0.0459 ± 0.0038	1.2500
Hyperlap	0.0924 ± 0.0091	0.0067 ± 0.0013	0.1079 ± 0.0043	0.1084 ± 0.0042	2.5000
HyperPLR	0.0569 ± 0.0044	0.1864 ± 0.0085	0.1863 ± 0.0033	0.1869 ± 0.0048	3.2500
Hyperlap+	0.3728 ± 0.0012	0.2249 ± 0.0020	0.0686 ± 0.0042	0.0727 ± 0.0030	3.5000
HyperDK00	0.3260 ± 0.0006	0.4256 ± 0.0056	0.3768 ± 0.0006	0.3777 ± 0.0007	5.2500
HyperDK11	0.3327 ± 0.0012	0.4420 ± 0.0083	0.3635 ± 0.0002	0.3649 ± 0.0004	5.2500

Table 26: **Macroscopic structure** performance comparison on the **NDC-classes** dataset. Methods are grouped by kind. Reported metrics: Incidence Matrix Density (lower is better), LCC Diameter (lower is better), Clique Exp. # Edges (lower is better), Line Graph # Edges (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Incidence Matrix Density	LCC Diameter	Clique Exp. # Edges	Line Graph # Edges	Avg. Rank
unconstrained					
Janus	0.0046 ± 0.0006	0.8000 ± 0.4472	159.0000 ± 142.0194	$82433.8000 \pm 15422.4404$	1.5000
TheRA	0.0004	2.0000	10166.0000	25451.0000	2.2500
SubsetSampling	0.0046 ± 0.0006	4.0000 ± 0.0000	4380.8000 ± 646.1391	$97821.0000 \pm 40655.9604$	3.0000
HyperPA	0.0049 ± 0.0003	5.0000 ± 0.0000	1851.4000 ± 367.2285	$214178.8000 \pm 62837.8174$	3.8750
NaivePA	0.0063 ± 0.0002	5.0000 ± 0.0000	5856.8000 ± 914.3502	$180947.8000 \pm 53300.6257$	4.3750
(Janus) reconstruction upper bound					
JanusNC-β-VAE	0.0000 ± 0.0000	0.0000 ± 0.0000	5.4000 ± 5.5946	375.4000 ± 380.2714	1.0000
Janus-β-VAE	0.0000 ± 0.0000	0.4000 ± 0.8944	54.2000 ± 22.5765	2107.4000 ± 279.9898	2.0000
node-set constrained					
JanusNC	0.0012 ± 0.0001	0.2000 ± 0.4472	98.6000 ± 6.5803	11413.0000 ± 751.0310	2.2500
HyperPLR	0.0010 ± 0.0002	0.0000 ± 0.0000	706.4000 ± 7.4364	34292.2000 ± 4907.9905	2.5000
Hyperlap+	0.0014 ± 0.0000	2.6000 ± 0.5477	8998.6000 ± 280.6533	9905.6000 ± 3257.1133	3.2500
HyperDK00	0.0000 ± 0.0000	4.6000 ± 0.5477	12545.0000 ± 69.6671	18179.0000 ± 258.0949	3.7500
HyperDK11	0.0001 ± 0.0000	4.2000 ± 0.4472	12290.6000 ± 143.4932	71341.2000 ± 916.8335	4.2500
Hyperlap	0.0013 ± 0.0001	3.8000 ± 0.4472	13230.2000 ± 291.4107	35283.8000 ± 837.5964	5.0000

value. Lower values indicate better preservation of global community structure, and the average rank aggregates the modularity-based metrics.

F.5 Macroscopic structure

Tables 26, 27, 28, 29, and 30 report macroscopic structural results for the NDC-classes, contact-high-school, contact-primary-school, email-Enron, and email-Eu datasets, respectively. These tables evaluate large-scale structural properties of the generated hypergraphs. The reported metrics are incidence-matrix density, largest-connected-component diameter, the number of edges in the clique expansion, and the number of edges in the line graph, each reported as an error relative to the real dataset. Lower values indicate better preservation of the global structure, and the average rank summarizes performance across these metrics.

Table 27: **Macroscopic structure** performance comparison on the **contact-high-school** dataset. Methods are grouped by kind. Reported metrics: Incidence Matrix Density (lower is better), LCC Diameter (lower is better), Clique Exp. # Edges (lower is better), Line Graph # Edges (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Incidence Matrix Density	LCC Diameter	Clique Exp. # Edges	Line Graph # Edges	Avg. Rank
unconstrained					
Janus	0.0000 ± 0.0000	0.8000 ± 0.4472	391.6000 ± 35.1184	24882.0000 ± 5694.4898	1.8750
HyperPA	0.0002 ± 0.0001	0.0000 ± 0.0000	616.4000 ± 248.7877	734993.2000 ± 70823.9485	2.7500
SubsetSampling	0.0001 ± 0.0001	0.2000 ± 0.4472	1100.0000 ± 220.3497	540273.2000 ± 73085.8941	2.7500
TheRA	0.0009	1.0000	603.0000	12563.0000	3.0000
NaivePA	0.0033 ± 0.0001	0.8000 ± 0.4472	6396.4000 ± 442.5306	2613255.8000 ± 334047.7548	4.6250
(Janus) reconstruction upper bound					
JanusNC-β-VAE	0.0000 ± 0.0000	0.0000 ± 0.0000	2.8000 ± 2.1679	2346.2000 ± 439.4271	1.1250
Janus-β-VAE	0.0000 ± 0.0000	0.0000 ± 0.0000	21.0000 ± 2.0000	10152.4000 ± 1683.2713	1.8750
node-set constrained					
JanusNC	0.0001 ± 0.0000	0.0000 ± 0.0000	141.0000 ± 6.8191	4666.0000 ± 502.5221	1.7500
Hyperlap+	0.0000 ± 0.0000	1.0000 ± 0.0000	1218.8000 ± 43.8030	13909.2000 ± 2474.6313	2.6250
Hyperlap	0.0000 ± 0.0000	0.8000 ± 0.4472	2865.2000 ± 49.1650	16890.4000 ± 2438.8031	2.8750
HyperDK11	0.0000 ± 0.0000	0.8000 ± 0.4472	5217.6000 ± 29.7456	42017.2000 ± 1479.0841	3.6250
HyperPLR	0.0012 ± 0.0000	1.0000 ± 0.0000	829.6000 ± 5.6833	338960.4000 ± 5610.0259	4.3750
HyperDK00	0.0020 ± 0.0000	2.0000 ± 0.0000	11568.6000 ± 114.3451	176494.8000 ± 2118.9379	5.7500

Table 28: **Macroscopic structure** performance comparison on the **contact-primary-school** dataset. Methods are grouped by kind. Reported metrics: Incidence Matrix Density (lower is better), LCC Diameter (lower is better), Clique Exp. # Edges (lower is better), Line Graph # Edges (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Incidence Matrix Density	LCC Diameter	Clique Exp. # Edges	Line Graph # Edges	Avg. Rank
unconstrained					
Janus	0.0000 ± 0.0000	0.2000 ± 0.4472	339.8000 ± 22.6208	79106.8000 ± 8182.4088	1.8750
TheRA	0.0014	0.0000	714.0000	326828.0000	2.7500
HyperPA	0.0001 ± 0.0000	0.2000 ± 0.4472	580.0000 ± 419.0692	1606982.2000 ± 443553.1759	2.8750
SubsetSampling	0.0001 ± 0.0000	0.0000 ± 0.0000	829.4000 ± 474.7924	1985241.6000 ± 784304.1783	3.2500
NaivePA	0.0042 ± 0.0000	0.0000 ± 0.0000	4214.4000 ± 624.7234	7413493.4000 ± 1293515.7968	4.2500
(Janus) reconstruction upper bound					
JanusNC-β-VAE	0.0000 ± 0.0000	0.0000 ± 0.0000	2.0000 ± 1.4142	6485.8000 ± 2344.5962	1.1250
Janus-β-VAE	0.0000 ± 0.0000	0.0000 ± 0.0000	26.8000 ± 5.0200	34578.2000 ± 2824.2333	1.8750
node-set constrained					
JanusNC	0.0000 ± 0.0000	0.0000 ± 0.0000	143.8000 ± 12.1943	11737.0000 ± 1662.6089	1.8750
Hyperlap	0.0000 ± 0.0000	0.2000 ± 0.4472	1934.0000 ± 62.1410	7266.8000 ± 3264.9594	2.3750
Hyperlap+	0.0000 ± 0.0000	1.0000 ± 0.0000	263.4000 ± 214.5001	67957.6000 ± 21124.7408	2.8750
HyperPLR	0.0010 ± 0.0000	0.0000 ± 0.0000	1455.0000 ± 0.0000	622811.4000 ± 31925.1994	3.6250
HyperDK11	0.0001 ± 0.0000	1.0000 ± 0.0000	6388.8000 ± 29.3462	357168.2000 ± 9569.1507	4.5000
HyperDK00	0.0029 ± 0.0000	1.0000 ± 0.0000	12522.0000 ± 77.9519	676662.8000 ± 5594.9212	5.7500

Table 29: **Macroscopic structure** performance comparison on the **email-Enron** dataset. Methods are grouped by kind. Reported metrics: Incidence Matrix Density (lower is better), LCC Diameter (lower is better), Clique Exp. # Edges (lower is better), Line Graph # Edges (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Incidence Matrix Density	LCC Diameter	Clique Exp. # Edges	Line Graph # Edges	Avg. Rank
unconstrained					
Janus	0.0021 ± 0.0003	0.0000 ± 0.0000	13.2000 ± 10.5451	20514.4000 ± 3063.9399	1.8750
HyperPA	0.0018 ± 0.0010	0.0000 ± 0.0000	244.4000 ± 154.3869	$31887.8000 \pm 26026.0117$	2.1250
SubsetSampling	0.0017 ± 0.0006	0.2000 ± 0.4472	431.0000 ± 107.2357	$41643.8000 \pm 11717.4138$	2.7500
TheRA	0.0026	1.0000	1275.0000	14821.0000	3.6250
NaivePA	0.0098 ± 0.0008	1.0000 ± 0.0000	1038.8000 ± 178.2939	$112411.2000 \pm 54623.5179$	4.6250
(Janus) reconstruction upper bound					
JanusNC-β-VAE	0.0000 ± 0.0000	0.0000 ± 0.0000	2.0000 ± 4.4721	266.4000 ± 332.3316	1.2500
Janus-β-VAE	0.0000 ± 0.0000	0.0000 ± 0.0000	2.0000 ± 2.1213	461.8000 ± 388.9758	1.7500
node-set constrained					
JanusNC	0.0003 ± 0.0001	0.0000 ± 0.0000	15.0000 ± 4.6368	2228.2000 ± 821.6256	1.3750
Hyperlap+	0.0009 ± 0.0000	0.2000 ± 0.4472	686.0000 ± 35.3483	3603.8000 ± 1964.1286	3.0000
HyperDK11	0.0001 ± 0.0001	1.0000 ± 0.0000	2161.2000 ± 22.6539	5507.4000 ± 878.6420	3.6250
HyperPLR	0.0020 ± 0.0002	0.0000 ± 0.0000	62.4000 ± 3.3615	23527.8000 ± 3819.9611	3.6250
Hyperlap	0.0009 ± 0.0001	1.0000 ± 0.0000	1846.4000 ± 106.7183	2849.4000 ± 1528.9669	3.6250
HyperDK00	0.0020 ± 0.0001	2.0000 ± 0.0000	3064.8000 ± 28.3231	22566.4000 ± 422.1194	5.7500

Table 30: **Macroscopic structure** performance comparison on the **email-Eu** dataset. Methods are grouped by kind. Reported metrics: Incidence Matrix Density (lower is better), LCC Diameter (lower is better), Clique Exp. # Edges (lower is better), Line Graph # Edges (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Incidence Matrix Density	LCC Diameter	Clique Exp. # Edges	Line Graph # Edges	Avg. Rank
unconstrained					
SubsetSampling	0.0000 ± 0.0000	2.0000 ± 0.0000	28833.4000 ± 3498.3531	$10460547.0000 \pm 2901237.0644$	2.0000
Janus	0.0023 ± 0.0000	0.6667 ± 0.5774	16744.6667 ± 248.4056	$22185259.0000 \pm 258695.8816$	2.7500
HyperPA	0.0003 ± 0.0003	2.2000 ± 0.4472	10587.2000 ± 1468.2138	$34082994.6000 \pm 3768618.3128$	2.8750
TheRA	0.0013	4.0000	32109.0000	356438.0000	3.5000
NaivePA	0.0010 ± 0.0000	2.2000 ± 0.4472	36980.0000 ± 1142.5067	$30468155.6000 \pm 5423212.4123$	3.8750
(Janus) reconstruction upper bound					
JanusNC-β-VAE	0.0000 ± 0.0000	0.0000 ± 0.0000	57.2000 ± 19.1233	$123360.2000 \pm 18285.6379$	1.1250
Janus-β-VAE	0.0001 ± 0.0000	0.0000 ± 0.0000	755.0000 ± 69.9393	$700513.2000 \pm 31178.1844$	1.8750
node-set constrained					
HyperPLR	0.0002 ± 0.0000	0.2000 ± 0.4472	13615.0000 ± 88.9326	$1239056.4000 \pm 243479.8529$	2.2500
Hyperlap+	0.0004 ± 0.0000	0.8000 ± 0.4472	20184.4000 ± 306.8897	$165310.4000 \pm 28866.5000$	3.0000
HyperDK11	0.0001 ± 0.0000	2.0000 ± 0.0000	69432.0000 ± 284.6621	$470975.4000 \pm 31351.5873$	3.1250
JanusNC	0.0009 ± 0.0000	0.4000 ± 0.5477	2618.0000 ± 155.0048	$3887615.2000 \pm 94701.3252$	3.5000
Hyperlap	0.0004 ± 0.0000	2.0000 ± 0.0000	57953.8000 ± 1494.2255	$769716.6000 \pm 85962.1059$	3.8750
HyperDK00	0.0003 ± 0.0000	4.0000 ± 0.0000	97434.4000 ± 438.4140	$4874286.4000 \pm 3305.5184$	5.2500

Table 31: **HyperTrans and Overlapness** performance comparison on the **NDC-classes** dataset. Methods are grouped by kind. Reported metrics: HyperTrans (lower is better), Global Overlapness (lower is better), Egonet Overlapness (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	HyperTrans	Global Overlapness	Egonet Overlapness	Avg. Rank
unconstrained				
TheRA	0.0002	0.7272	0.3862	1.3333
HyperPA	0.0111 ± 0.0043	5.1861 ± 1.1490	0.3267 ± 0.0761	2.6667
Janus	0.0073 ± 0.0043	5.0093 ± 0.6927	0.5747 ± 0.1380	3.3333
SubsetSampling	0.0338 ± 0.0045	4.6436 ± 0.9122	0.4081 ± 0.0463	3.3333
NaivePA	0.0279 ± 0.0041	6.5544 ± 0.9227	0.5072 ± 0.0331	4.3333
(Janus) reconstruction upper bound				
JanusNC-β-VAE	0.0002 ± 0.0001	0.0215 ± 0.0149	0.0056 ± 0.0038	1.0000
Janus-β-VAE	0.0006 ± 0.0005	0.1602 ± 0.0184	0.0422 ± 0.0041	2.0000
node-set constrained				
HyperPLR	0.0354 ± 0.0078	1.1054 ± 0.1668	0.2889 ± 0.0498	2.3333
JanusNC	0.0043 ± 0.0016	1.3110 ± 0.0678	0.3133 ± 0.0279	3.0000
Hyperlap+	0.0211 ± 0.0036	1.2179 ± 0.0511	0.4803 ± 0.0388	3.3333
HyperDK00	0.0511 ± 0.0000	0.0166 ± 0.0130	0.8824 ± 0.0016	4.0000
Hyperlap	0.0416 ± 0.0008	1.1680 ± 0.0939	0.7485 ± 0.0143	4.0000
HyperDK11	0.0440 ± 0.0004	0.1874 ± 0.0119	0.8916 ± 0.0018	4.3333

Table 32: **HyperTrans and Overlapness** performance comparison on the **contact-high-school** dataset. Methods are grouped by kind. Reported metrics: HyperTrans (lower is better), Global Overlapness (lower is better), Egonet Overlapness (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	HyperTrans	Global Overlapness	Egonet Overlapness	Avg. Rank
unconstrained				
Janus	0.0280 ± 0.0027	0.2615 ± 0.0763	0.1879 ± 0.0152	1.3333
HyperPA	0.2821 ± 0.0056	2.0974 ± 1.8201	0.9480 ± 0.0328	3.0000
TheRA	0.0014	6.7492	2.0610	3.0000
SubsetSampling	0.1073 ± 0.0134	8.0551 ± 1.1319	1.3749 ± 0.0112	3.6667
NaivePA	0.2527 ± 0.0038	19.8167 ± 2.7867	1.1368 ± 0.0524	4.0000
(Janus) reconstruction upper bound				
JanusNC-β-VAE	0.0005 ± 0.0001	0.1058 ± 0.0195	0.0061 ± 0.0018	1.0000
Janus-β-VAE	0.0023 ± 0.0004	0.4606 ± 0.0672	0.0282 ± 0.0060	2.0000
node-set constrained				
Hyperlap+	0.0249 ± 0.0015	0.0000 ± 0.0000	0.6173 ± 0.0178	1.6667
JanusNC	0.0075 ± 0.0010	0.5020 ± 0.0918	0.0302 ± 0.0040	1.6667
Hyperlap	0.1665 ± 0.0018	0.0683 ± 0.0935	1.1299 ± 0.0128	2.6667
HyperDK11	0.2352 ± 0.0010	2.5694 ± 0.0745	1.6703 ± 0.0046	4.6667
HyperPLR	0.2109 ± 0.0029	9.3780 ± 0.0729	1.6658 ± 0.0168	4.6667
HyperDK00	0.3173 ± 0.0007	4.8538 ± 0.1029	1.8636 ± 0.0063	5.6667

F.5.1 HyperTrans and Overlapness

Tables 31, 32, 33, 34, and 35 report macroscopic transitivity and overlapness results for the NDC-classes, contact-high-school, contact-primary-school, email-Enron, and email-Eu datasets, respectively. These tables focus on higher-order global interaction patterns that are specific to hypergraphs. The reported metrics are hypergraph transitivity, global overlapness, and egonet overlapness, measured as errors with respect to the corresponding real-hypergraph values. Lower values indicate that the generated hypergraphs better match the real transitivity and hyperedge-overlap structure, while the average rank summarizes the combined performance.

Table 33: **HyperTrans and Overlapness** performance comparison on the **contact-primary-school** dataset. Methods are grouped by kind. Reported metrics: HyperTrans (lower is better), Global Overlapness (lower is better), Egonet Overlapness (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	HyperTrans	Global Overlapness	Egonet Overlapness	Avg. Rank
unconstrained				
Janus	0.0136 ± 0.0023	0.3316 ± 0.1783	0.1265 ± 0.0211	1.3333
HyperPA	0.1999 ± 0.0038	10.1188 ± 6.1790	1.0053 ± 0.0948	3.0000
TheRA	0.0024	18.2727	2.7430	3.0000
SubsetSampling	0.0202 ± 0.0050	22.2888 ± 7.6265	1.4931 ± 0.0780	3.6667
NaivePA	0.2036 ± 0.0085	34.7454 ± 8.3622	0.5792 ± 0.1201	4.0000
(Janus) reconstruction upper bound				
JanusNC-β-VAE	0.0002 ± 0.0001	0.1810 ± 0.0486	0.0064 ± 0.0011	1.0000
Janus-β-VAE	0.0018 ± 0.0002	1.0091 ± 0.0713	0.0346 ± 0.0049	2.0000
node-set constrained				
JanusNC	0.0091 ± 0.0006	0.5702 ± 0.0316	0.0537 ± 0.0070	1.6667
Hyperlap+	0.0119 ± 0.0088	0.0000 ± 0.0000	0.1621 ± 0.1541	1.8333
Hyperlap	0.0558 ± 0.0019	0.0000 ± 0.0000	0.9482 ± 0.0193	2.5000
HyperDK11	0.1293 ± 0.0012	10.7215 ± 0.2688	2.1957 ± 0.0035	4.6667
HyperPLR	0.1316 ± 0.0019	12.6132 ± 0.2169	1.9294 ± 0.0252	5.0000
HyperDK00	0.2671 ± 0.0006	12.5322 ± 0.1986	2.1645 ± 0.0062	5.3333

Table 34: **HyperTrans and Overlapness** performance comparison on the **email-Enron** dataset. Methods are grouped by kind. Reported metrics: HyperTrans (lower is better), Global Overlapness (lower is better), Egonet Overlapness (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	HyperTrans	Global Overlapness	Egonet Overlapness	Avg. Rank
unconstrained				
TheRA	0.0043	2.7273	0.3241	1.0000
Janus	0.0296 ± 0.0038	3.1622 ± 0.3972	1.0059 ± 0.1739	2.3333
HyperPA	0.1053 ± 0.0070	3.3671 ± 2.2879	1.7273 ± 0.1088	3.3333
SubsetSampling	0.0210 ± 0.0123	6.1716 ± 1.0520	2.2729 ± 0.0614	4.0000
NaivePA	0.1177 ± 0.0091	5.0146 ± 3.3976	1.8691 ± 0.1608	4.3333
(Janus) reconstruction upper bound				
JanusNC-β-VAE	0.0004 ± 0.0004	0.0448 ± 0.0607	0.0034 ± 0.0036	1.0000
Janus-β-VAE	0.0004 ± 0.0005	0.0937 ± 0.0663	0.0117 ± 0.0097	2.0000
node-set constrained				
Hyperlap+	0.0301 ± 0.0037	0.1259 ± 0.0000	0.6274 ± 0.0378	1.6667
JanusNC	0.0044 ± 0.0023	0.3885 ± 0.1201	0.1291 ± 0.0322	1.6667
Hyperlap	0.0813 ± 0.0023	0.2159 ± 0.1232	1.9986 ± 0.0860	3.0000
HyperPLR	0.0848 ± 0.0035	2.9634 ± 0.3778	0.7244 ± 0.1058	4.3333
HyperDK11	0.1198 ± 0.0018	1.5804 ± 0.1058	2.7366 ± 0.0137	5.0000
HyperDK00	0.1690 ± 0.0006	1.1622 ± 0.1045	2.9575 ± 0.0123	5.3333

Table 35: **HyperTrans and Overlapness** performance comparison on the **email-Eu** dataset. Methods are grouped by kind. Reported metrics: HyperTrans (lower is better), Global Overlapness (lower is better), Egonet Overlapness (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	HyperTrans	Global Overlapness	Egonet Overlapness	Avg. Rank
unconstrained				
HyperPA	0.0211 ± 0.0022	4.2416 ± 2.1802	2.5797 ± 0.1585	2.0000
TheRA	0.0041	30.8833	2.1682	2.3333
Janus	0.0637 ± 0.0012	56.6115 ± 1.1946	0.8180 ± 0.0539	3.3333
SubsetSampling	0.0502 ± 0.0086	4.3924 ± 2.7291	3.8669 ± 0.0598	3.3333
NaivePA	0.0713 ± 0.0053	17.9941 ± 2.4858	3.7265 ± 0.0494	4.0000
(Janus) reconstruction upper bound				
JanusNC-β-VAE	0.0008 ± 0.0002	0.6580 ± 0.1080	0.0585 ± 0.0142	1.0000
Janus-β-VAE	0.0047 ± 0.0003	3.8457 ± 0.2356	0.3265 ± 0.0340	2.0000
node-set constrained				
HyperPLR	0.0720 ± 0.0004	5.2459 ± 0.5397	0.0619 ± 0.0462	2.6667
Hyperlap+	0.0146 ± 0.0026	7.7240 ± 0.4500	1.6382 ± 0.0201	3.0000
JanusNC	0.0030 ± 0.0007	23.4498 ± 0.5599	3.4109 ± 0.0709	3.3333
Hyperlap	0.0525 ± 0.0009	7.4287 ± 0.4497	3.5670 ± 0.0414	3.6667
HyperDK11	0.0778 ± 0.0003	2.9966 ± 0.1062	4.3552 ± 0.0047	4.0000
HyperDK00	0.1195 ± 0.0001	2.0727 ± 0.0529	4.6688 ± 0.0015	4.3333

Table 36: **Macroscopic size** performance comparison on the **NDC-classes** dataset. Methods are grouped by kind. Reported metrics: # Nodes (lower is better), # Hyperedges (lower is better), LCC # Nodes (lower is better), LCC # Hyperedges (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	# Nodes	# Hyperedges	LCC # Nodes	LCC # Hyperedges	Avg. Rank
unconstrained					
Janus	442.2000 ± 20.4377	0.0000 ± 0.0000	20.8000 ± 10.3296	193.2000 ± 9.8336	1.2500
SubsetSampling	554.4000 ± 21.1258	83.0000 ± 46.5403	26.2000 ± 21.8792	229.8000 ± 95.3950	2.7500
TheRA	12.0000	60.0000	495.0000	306.0000	3.2500
NaivePA	560.4000 ± 17.7285	82.6000 ± 24.6840	27.4000 ± 17.7285	249.4000 ± 92.1916	3.5000
HyperPA	568.2000 ± 18.9526	92.8000 ± 67.9389	50.4000 ± 15.5338	246.8000 ± 118.1194	4.2500
(Janus) reconstruction upper bound					
JanusNC-β-VAE	0.0000 ± 0.0000	4.2000 ± 2.3875	2.0000 ± 3.9370	5.0000 ± 4.0620	1.1250
Janus-β-VAE	0.0000 ± 0.0000	25.0000 ± 4.8477	35.2000 ± 39.2326	37.4000 ± 20.0823	1.8750
node-set constrained					
JanusNC	139.4000 ± 3.5071	0.0000 ± 0.0000	20.0000 ± 4.3589	51.8000 ± 3.4928	1.8750
HyperPLR	261.4000 ± 11.0136	0.0000 ± 0.0000	19.4000 ± 2.0736	126.0000 ± 10.3682	2.6250
HyperDK11	0.0000 ± 0.0000	10.6000 ± 2.7019	527.6000 ± 3.5777	256.8000 ± 3.7014	3.8750
HyperDK00	5.4000 ± 3.3615	2.0000 ± 1.5811	527.6000 ± 3.3615	270.0000 ± 1.5811	4.1250
Hyperlap	198.0000 ± 13.2853	39.0000 ± 0.0000	335.0000 ± 13.2853	233.0000 ± 0.0000	4.2500
Hyperlap+	205.2000 ± 7.1903	39.0000 ± 0.0000	327.8000 ± 7.1903	233.0000 ± 0.0000	4.2500

F.6 Macroscopic size

Tables 36, 37, 38, 39, and 40 report macroscopic size statistics for the NDC-classes, contact-high-school, contact-primary-school, email-Enron, and email-Eu datasets, respectively. These tables evaluate whether each method produces hypergraphs with the correct global scale. The reported metrics are the errors in the number of nodes, the number of hyperedges, the number of nodes in the largest connected component, and the number of hyperedges in the largest connected component. Lower values indicate better agreement with the size and connectivity profile of the real hypergraph, and the average rank summarizes performance across the size metrics.

Table 37: **Macroscopic size** performance comparison on the **contact-high-school** dataset. Methods are grouped by kind. Reported metrics: # Nodes (lower is better), # Hyperedges (lower is better), LCC # Nodes (lower is better), LCC # Hyperedges (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	# Nodes	# Hyperedges	LCC # Nodes	LCC # Hyperedges	Avg. Rank
unconstrained					
TheRA	0.0000	0.0000	0.0000	0.0000	1.1250
Janus	0.2000 ± 0.4472	0.0000 ± 0.0000	1.0000 ± 1.0000	0.8000 ± 0.8367	1.8750
HyperPA	8.2000 ± 3.7683	342.4000 ± 309.4217	8.2000 ± 3.7683	342.4000 ± 309.4217	4.0000
NaivePA	6.8000 ± 2.6833	554.6000 ± 259.1135	6.8000 ± 2.6833	554.6000 ± 259.1135	4.0000
SubsetSampling	4.4000 ± 1.8166	1241.8000 ± 118.7800	4.4000 ± 1.8166	1241.8000 ± 118.7800	4.0000
(Janus) reconstruction upper bound					
JanusNC-β-VAE	0.0000 ± 0.0000	16.0000 ± 3.0822	0.0000 ± 0.0000	16.0000 ± 3.0822	1.2500
Janus-β-VAE	0.0000 ± 0.0000	46.8000 ± 4.4944	0.0000 ± 0.0000	46.8000 ± 4.4944	1.7500
node-set constrained					
Hyperlap+	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	2.2500
JanusNC	0.2000 ± 0.4472	0.0000 ± 0.0000	0.2000 ± 0.4472	0.0000 ± 0.0000	3.2500
HyperDK11	0.0000 ± 0.0000	395.4000 ± 11.0589	0.0000 ± 0.0000	395.4000 ± 11.0589	3.5000
Hyperlap	0.4000 ± 0.5477	0.0000 ± 0.0000	0.4000 ± 0.5477	0.0000 ± 0.0000	3.7500
HyperDK00	0.0000 ± 0.0000	2257.6000 ± 37.7995	0.0000 ± 0.0000	2257.6000 ± 37.7995	4.0000
HyperPLR	2.0000 ± 0.0000	0.0000 ± 0.0000	2.0000 ± 0.0000	0.0000 ± 0.0000	4.2500

Table 38: **Macroscopic size** performance comparison on the **contact-primary-school** dataset. Methods are grouped by kind. Reported metrics: # Nodes (lower is better), # Hyperedges (lower is better), LCC # Nodes (lower is better), LCC # Hyperedges (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	# Nodes	# Hyperedges	LCC # Nodes	LCC # Hyperedges	Avg. Rank
unconstrained					
TheRA	0.0000	0.0000	0.0000	0.0000	1.2500
Janus	0.2000 ± 0.4472	0.0000 ± 0.0000	0.2000 ± 0.4472	0.0000 ± 0.0000	1.7500
HyperPA	1.8000 ± 0.8367	1033.0000 ± 694.5265	1.8000 ± 0.8367	1033.0000 ± 694.5265	3.5000
NaivePA	1.4000 ± 0.8944	1314.0000 ± 586.8893	1.4000 ± 0.8944	1314.0000 ± 586.8893	3.5000
SubsetSampling	2.6000 ± 1.3416	2360.6000 ± 758.3576	2.6000 ± 1.3416	2360.6000 ± 758.3576	5.0000
(Janus) reconstruction upper bound					
JanusNC-β-VAE	0.0000 ± 0.0000	18.4000 ± 5.3666	0.0000 ± 0.0000	18.4000 ± 5.3666	1.2500
Janus-β-VAE	0.0000 ± 0.0000	80.0000 ± 5.8737	0.0000 ± 0.0000	80.0000 ± 5.8737	1.7500
node-set constrained					
Hyperlap	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	2.7500
Hyperlap+	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	2.7500
JanusNC	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	2.7500
HyperDK11	0.0000 ± 0.0000	1241.8000 ± 32.8283	0.0000 ± 0.0000	1241.8000 ± 32.8283	4.0000
HyperPLR	1.0000 ± 0.0000	0.0000 ± 0.0000	1.0000 ± 0.0000	0.0000 ± 0.0000	4.2500
HyperDK00	0.0000 ± 0.0000	3809.2000 ± 31.6259	0.0000 ± 0.0000	3809.2000 ± 31.6259	4.5000

Table 39: **Macroscopic size** performance comparison on the **email-Enron** dataset. Methods are grouped by kind. Reported metrics: # Nodes (lower is better), # Hyperedges (lower is better), LCC # Nodes (lower is better), LCC # Hyperedges (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	# Nodes	# Hyperedges	LCC # Nodes	LCC # Hyperedges	Avg. Rank
unconstrained					
Janus	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	1.2500
TheRA	0.0000	53.0000	0.0000	53.0000	1.7500
HyperPA	13.0000 ± 3.3912	253.0000 ± 116.8696	13.6000 ± 2.8810	253.6000 ± 116.1671	3.7500
SubsetSampling	8.8000 ± 2.1679	380.6000 ± 72.6726	9.0000 ± 2.2361	380.8000 ± 72.6684	4.0000
NaivePA	13.2000 ± 2.1679	316.0000 ± 114.5949	13.2000 ± 2.1679	316.0000 ± 114.5949	4.2500
(Janus) reconstruction upper bound					
JanusNC-β-VAE	0.0000 ± 0.0000	1.0000 ± 1.2247	0.0000 ± 0.0000	1.0000 ± 1.2247	1.2500
Janus-β-VAE	0.0000 ± 0.0000	3.0000 ± 2.0000	0.0000 ± 0.0000	3.0000 ± 2.0000	1.7500
node-set constrained					
Hyperlap+	0.0000 ± 0.0000	53.0000 ± 0.0000	0.0000 ± 0.0000	53.0000 ± 0.0000	2.7500
JanusNC	0.6000 ± 0.5477	0.0000 ± 0.0000	0.6000 ± 0.5477	0.0000 ± 0.0000	3.2500
HyperDK11	0.0000 ± 0.0000	82.6000 ± 8.2946	0.0000 ± 0.0000	82.6000 ± 8.2946	3.5000
HyperPLR	2.0000 ± 0.0000	0.0000 ± 0.0000	2.0000 ± 0.0000	0.0000 ± 0.0000	3.7500
Hyperlap	0.4000 ± 0.5477	53.0000 ± 0.0000	0.4000 ± 0.5477	53.0000 ± 0.0000	3.7500
HyperDK00	0.0000 ± 0.0000	183.6000 ± 11.3490	0.0000 ± 0.0000	183.6000 ± 11.3490	4.0000

Table 40: **Macroscopic size** performance comparison on the **email-Eu** dataset. Methods are grouped by kind. Reported metrics: # Nodes (lower is better), # Hyperedges (lower is better), LCC # Nodes (lower is better), LCC # Hyperedges (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	# Nodes	# Hyperedges	LCC # Nodes	LCC # Hyperedges	Avg. Rank
unconstrained					
Janus	12.3333 ± 7.0238	0.0000 ± 0.0000	6.3333 ± 6.5064	10.0000 ± 3.6056	1.7500
TheRA	12.0000	507.0000	7.0000	488.0000	2.2500
SubsetSampling	3.6000 ± 1.3416	1356.8000 ± 823.6171	14.6000 ± 2.0736	1338.6000 ± 823.6785	2.5000
NaivePA	3.8000 ± 2.0494	1656.0000 ± 527.4263	15.2000 ± 2.0494	1637.0000 ± 527.4263	3.5000
HyperPA	80.8000 ± 79.2950	2039.4000 ± 1234.3607	71.8000 ± 69.7044	2022.4000 ± 1233.1870	5.0000
(Janus) reconstruction upper bound					
JanusNC-β-VAE	0.2000 ± 0.4472	89.4000 ± 10.8995	0.0000 ± 0.0000	89.6000 ± 10.9453	1.2500
Janus-β-VAE	0.0000 ± 0.0000	492.6000 ± 20.5256	0.4000 ± 0.5477	493.0000 ± 20.5791	1.7500
node-set constrained					
JanusNC	47.8000 ± 4.0249	0.0000 ± 0.0000	35.8000 ± 5.4037	12.0000 ± 2.9155	2.8750
HyperDK11	0.0000 ± 0.0000	1378.4000 ± 43.1080	18.2000 ± 0.4472	1360.2000 ± 42.8567	3.1250
Hyperlap	46.8000 ± 4.6043	507.0000 ± 0.0000	27.8000 ± 4.6043	488.0000 ± 0.0000	3.2500
HyperDK00	0.0000 ± 0.0000	2712.4000 ± 93.5217	19.0000 ± 0.0000	2693.4000 ± 93.5217	3.8750
HyperPLR	137.4000 ± 1.5166	0.0000 ± 0.0000	118.4000 ± 1.5166	19.0000 ± 0.0000	3.8750
Hyperlap+	49.8000 ± 4.5497	507.0000 ± 0.0000	30.8000 ± 4.5497	488.0000 ± 0.0000	4.0000

Table 41: **Higher-order dissimilarity measures** performance comparison on the **NDC-classes** dataset. Methods are grouped by kind. Reported metrics: Hyper NetSimile (lower is better), Hyperedge Portrait Divergence (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Hyper NetSimile	Hyperedge Portrait Divergence	Avg. Rank
unconstrained			
Janus	6.0414 ± 0.3021	0.4210 ± 0.0484	1.5000
TheRA	9.7695	0.3320	2.0000
HyperPA	8.0762 ± 0.4667	0.6145 ± 0.0348	3.0000
SubsetSampling	9.8929 ± 0.4720	0.5203 ± 0.0553	3.5000
NaivePA	10.2800 ± 0.3795	0.7330 ± 0.0039	5.0000
(Janus) reconstruction upper bound			
JanusNC-β-VAE	0.4208 ± 0.5567	0.0217 ± 0.0295	1.0000
Janus-β-VAE	1.1361 ± 0.0679	0.0561 ± 0.0299	2.0000
node-set constrained			
JanusNC	2.5608 ± 0.0646	0.1518 ± 0.0259	1.0000
HyperPLR	3.3487 ± 0.1112	0.2818 ± 0.0394	2.0000
HyperDK11	8.5485 ± 0.0541	0.6143 ± 0.0151	3.5000
Hyperlap+	8.4143 ± 0.3084	0.6148 ± 0.0223	3.5000
Hyperlap	9.8272 ± 0.0645	0.6455 ± 0.0306	5.0000
HyperDK00	10.2463 ± 0.0391	0.7673 ± 0.0258	6.0000

Table 42: **Higher-order dissimilarity measures** performance comparison on the **contact-high-school** dataset. Methods are grouped by kind. Reported metrics: Hyper NetSimile (lower is better), Hyperedge Portrait Divergence (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Hyper NetSimile	Hyperedge Portrait Divergence	Avg. Rank
unconstrained			
Janus	0.9456 ± 0.1792	0.3405 ± 0.0034	1.5000
TheRA	2.8572	0.3232	1.5000
HyperPA	5.2400 ± 0.0917	0.6133 ± 0.0137	3.0000
SubsetSampling	6.4425 ± 0.1279	0.8202 ± 0.0198	4.0000
NaivePA	6.5915 ± 0.1751	0.8949 ± 0.0011	5.0000
(Janus) reconstruction upper bound			
JanusNC-β-VAE	0.0130 ± 0.0040	0.0738 ± 0.0165	1.0000
Janus-β-VAE	0.0720 ± 0.0199	0.1026 ± 0.0067	2.0000
node-set constrained			
JanusNC	0.1946 ± 0.0177	0.2280 ± 0.0329	1.0000
Hyperlap	2.9377 ± 0.0784	0.3532 ± 0.0045	2.5000
HyperPLR	2.0690 ± 0.0146	0.7258 ± 0.0012	3.5000
Hyperlap+	3.0168 ± 0.0883	0.5113 ± 0.0209	3.5000
HyperDK11	3.3553 ± 0.0422	0.5258 ± 0.0135	4.5000
HyperDK00	6.0711 ± 0.0307	0.8222 ± 0.0048	6.0000

Table 43: **Higher-order dissimilarity measures** performance comparison on the **contact-primary-school** dataset. Methods are grouped by kind. Reported metrics: Hyper NetSimile (lower is better), Hyperedge Portrait Divergence (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Hyper NetSimile	Hyperedge Portrait Divergence	Avg. Rank
unconstrained			
Janus	0.3685 ± 0.0322	0.2277 ± 0.0043	1.0000
HyperPA	4.0878 ± 0.1772	0.5031 ± 0.0319	2.5000
TheRA	2.0996	0.5405	2.5000
NaivePA	5.2683 ± 0.1044	0.8612 ± 0.0009	4.5000
SubsetSampling	5.8668 ± 0.0935	0.8108 ± 0.0121	4.5000
(Janus) reconstruction upper bound			
JanusNC-β-VAE	0.0128 ± 0.0028	0.0876 ± 0.0176	1.0000
Janus-β-VAE	0.0661 ± 0.0144	0.1326 ± 0.0081	2.0000
node-set constrained			
JanusNC	0.1764 ± 0.0171	0.2020 ± 0.0031	1.0000
Hyperlap	2.1658 ± 0.0354	0.2114 ± 0.0031	2.5000
HyperPLR	1.6567 ± 0.0515	0.5826 ± 0.0023	3.5000
HyperDK11	3.5480 ± 0.0880	0.4578 ± 0.0046	4.0000
Hyperlap+	2.9876 ± 0.0155	0.5179 ± 0.0087	4.0000
HyperDK00	7.6628 ± 0.0568	0.8482 ± 0.0071	6.0000

Table 44: **Higher-order dissimilarity measures** performance comparison on the **email-Enron** dataset. Methods are grouped by kind. Reported metrics: Hyper NetSimile (lower is better), Hyperedge Portrait Divergence (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Hyper NetSimile	Hyperedge Portrait Divergence	Avg. Rank
unconstrained			
Janus	0.8984 ± 0.0678	0.1158 ± 0.0086	1.0000
HyperPA	3.8839 ± 0.2596	0.2858 ± 0.0575	3.0000
TheRA	3.5654	0.5158	3.0000
NaivePA	3.8490 ± 0.4478	0.6531 ± 0.0097	4.0000
SubsetSampling	4.1982 ± 0.1851	0.4526 ± 0.0382	4.0000
(Janus) reconstruction upper bound			
JanusNC-β-VAE	0.0125 ± 0.0216	0.0029 ± 0.0046	1.0000
Janus-β-VAE	0.0263 ± 0.0239	0.0073 ± 0.0063	2.0000
node-set constrained			
JanusNC	0.1391 ± 0.0181	0.0461 ± 0.0031	1.0000
HyperPLR	0.6153 ± 0.0898	0.2680 ± 0.0090	2.0000
HyperDK11	3.2826 ± 0.1244	0.4482 ± 0.0370	3.0000
Hyperlap+	3.5464 ± 0.3430	0.5310 ± 0.0259	4.0000
Hyperlap	3.7188 ± 0.1171	0.5543 ± 0.0277	5.0000
HyperDK00	6.7569 ± 0.0927	0.7562 ± 0.0263	6.0000

Table 45: **Higher-order dissimilarity measures** performance comparison on the **email-Eu** dataset. Methods are grouped by kind. Reported metrics: Hyper NetSimile (lower is better), Hyperedge Portrait Divergence (lower is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Hyper NetSimile	Hyperedge Portrait Divergence	Avg. Rank
unconstrained			
Janus	4.1857 ± 0.0573	0.4227 ± 0.0470	1.0000
SubsetSampling	4.6115 ± 0.3025	0.4287 ± 0.0390	2.0000
TheRA	6.1181	0.7134	3.5000
HyperPA	7.9956 ± 0.2381	0.5949 ± 0.0143	4.0000
NaivePA	6.2691 ± 0.2019	0.7350 ± 0.0033	4.5000
(Janus) reconstruction upper bound			
JanusNC-β-VAE	0.0387 ± 0.0078	0.0574 ± 0.0341	1.0000
Janus-β-VAE	0.3138 ± 0.0268	0.1125 ± 0.0023	2.0000
node-set constrained			
JanusNC	1.2224 ± 0.0476	0.3009 ± 0.0305	1.0000
HyperDK11	4.4075 ± 0.0247	0.4309 ± 0.0036	2.0000
Hyperlap	4.6837 ± 0.1403	0.5742 ± 0.0388	3.0000
Hyperlap+	4.7418 ± 0.0824	0.6210 ± 0.0114	4.0000
HyperPLR	4.8988 ± 0.0219	0.7806 ± 0.0097	5.0000
HyperDK00	9.8383 ± 0.0197	0.8432 ± 0.0032	6.0000

Table 46: **Reconstruction** performance comparison on the **NDC-classes** dataset. Methods are grouped by kind. Reported metrics: Intersection (higher is better), Jaccard Coefficient (higher is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Intersection	Jaccard Coefficient	Avg. Rank
(Janus) reconstruction upper bound			
JanusNC-β-VAE	1088.0000 ± 0.0000	0.9962 ± 0.0022	1.0000
Janus-β-VAE	1087.6000 ± 0.5477	0.9768 ± 0.0035	2.0000
node-set constrained			
JanusNC	971.8000 ± 4.9193	0.8070 ± 0.0074	1.0000
HyperPLR	348.4000 ± 10.0648	0.1907 ± 0.0065	2.0000
HyperDK11	6.4000 ± 1.8166	0.0030 ± 0.0008	3.0000
Hyperlap+	5.0000 ± 2.3452	0.0024 ± 0.0011	4.0000
Hyperlap	3.4000 ± 1.3416	0.0016 ± 0.0006	5.0000
HyperDK00	2.2000 ± 0.8367	0.0010 ± 0.0004	6.0000

F.7 Higher-order dissimilarities

Tables 41, 42, 43, 44, and 45 report higher-order dissimilarity results for the NDC-classes, contact-high-school, contact-primary-school, email-Enron, and email-Eu datasets, respectively. These tables assess empirical similarity between real and generated hypergraphs using hypergraph-level comparison measures rather than individual structural statistics. The reported metrics are Hyper NetSimile and Hyperedge Portrait Divergence, both of which measure dissimilarity between the generated and real hypergraphs. Lower values indicate that generated samples are more similar to the real data in terms of higher-order structure, and the average rank summarizes performance across the two dissimilarity measures.

F.8 Reconstruction

Tables 46, 47, 48, 49, and 50 report node-set constrained reconstruction results for the NDC-classes, contact-high-school, contact-primary-school, email-Enron, and email-Eu datasets, respectively. These tables evaluate how accurately each method recovers the observed hyperedge set when the node set is fixed. The reported metrics are the number of correctly recovered hyperedges, reported as

Table 47: **Reconstruction** performance comparison on the **contact-high-school** dataset. Methods are grouped by kind. Reported metrics: Intersection (higher is better), Jaccard Coefficient (higher is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Intersection	Jaccard Coefficient	Avg. Rank
(Janus) reconstruction upper bound			
JanusNC-β-VAE	7818.0000 ± 0.0000	0.9980 ± 0.0004	1.2500
Janus-β-VAE	7818.0000 ± 0.0000	0.9940 ± 0.0006	1.7500
node-set constrained			
JanusNC	7616.8000 ± 7.3280	0.9498 ± 0.0018	1.0000
HyperPLR	2765.8000 ± 31.6654	0.2149 ± 0.0030	2.0000
HyperDK11	574.8000 ± 31.0596	0.0392 ± 0.0022	3.0000
Hyperlap	494.6000 ± 24.0687	0.0352 ± 0.0018	4.0000
Hyperlap+	463.4000 ± 34.1877	0.0338 ± 0.0026	5.0000
HyperDK00	225.0000 ± 9.8742	0.0171 ± 0.0007	6.0000

Table 48: **Reconstruction** performance comparison on the **contact-primary-school** dataset. Methods are grouped by kind. Reported metrics: Intersection (higher is better), Jaccard Coefficient (higher is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Intersection	Jaccard Coefficient	Avg. Rank
(Janus) reconstruction upper bound			
JanusNC-β-VAE	12704.0000 ± 0.0000	0.9986 ± 0.0004	1.2500
Janus-β-VAE	12704.0000 ± 0.0000	0.9937 ± 0.0005	1.7500
node-set constrained			
JanusNC	12243.2000 ± 20.8014	0.9300 ± 0.0030	1.0000
HyperPLR	5127.0000 ± 50.2643	0.2528 ± 0.0031	2.0000
HyperDK11	1780.8000 ± 81.0876	0.0796 ± 0.0039	3.0000
Hyperlap	1390.0000 ± 21.7141	0.0657 ± 0.0011	4.0000
Hyperlap+	1227.2000 ± 69.4421	0.0608 ± 0.0035	5.0000
HyperDK00	857.4000 ± 14.3108	0.0413 ± 0.0007	6.0000

Table 49: **Reconstruction** performance comparison on the **email-Enron** dataset. Methods are grouped by kind. Reported metrics: Intersection (higher is better), Jaccard Coefficient (higher is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Intersection	Jaccard Coefficient	Avg. Rank
(Janus) reconstruction upper bound			
JanusNC-β-VAE	1512.0000 ± 0.0000	0.9993 ± 0.0008	1.2500
Janus-β-VAE	1512.0000 ± 0.0000	0.9980 ± 0.0013	1.7500
node-set constrained			
JanusNC	1450.2000 ± 4.8683	0.9215 ± 0.0060	1.0000
HyperPLR	419.8000 ± 19.0316	0.1613 ± 0.0085	2.0000
HyperDK11	82.2000 ± 5.0695	0.0287 ± 0.0018	3.0000
HyperDK00	74.8000 ± 5.9330	0.0270 ± 0.0022	4.0000
Hyperlap	53.2000 ± 6.0581	0.0194 ± 0.0023	5.0000
Hyperlap+	47.8000 ± 7.9812	0.0178 ± 0.0030	6.0000

Table 50: **Reconstruction** performance comparison on the **email-Eu** dataset. Methods are grouped by kind. Reported metrics: Intersection (higher is better), Jaccard Coefficient (higher is better). Colored cells mark the top three values within each kind for ranked metrics (green best, yellow second, orange third). Lower Avg. Rank is better.

Model	Intersection	Jaccard Coefficient	Avg. Rank
(Janus) reconstruction upper bound			
JanusNC-β-VAE	25005.0000 ± 3.6742	0.9947 ± 0.0003	1.0000
Janus-β-VAE	24983.6000 ± 2.0736	0.9773 ± 0.0008	2.0000
node-set constrained			
JanusNC	14693.4000 ± 38.7595	0.4155 ± 0.0016	1.0000
HyperPLR	3297.2000 ± 85.4939	0.0705 ± 0.0020	2.0000
HyperDK11	1022.4000 ± 66.8566	0.0215 ± 0.0014	3.0000
HyperDK00	715.6000 ± 11.7175	0.0153 ± 0.0002	4.0000
Hyperlap	652.2000 ± 40.7087	0.0140 ± 0.0009	5.0000
Hyperlap+	572.6000 ± 17.5585	0.0127 ± 0.0004	6.0000

Intersection, and the Jaccard coefficient between the generated and real hyperedge sets. Higher values indicate better reconstruction, while the average rank summarizes the reconstruction quality across the two metrics.

F.9 Comparison with DGMH details

Table 51: Comparison results using the DGMH evaluation framework on **NDC-classes** and **email-Eu** datasets.

Dataset	Metric	Motif	Real	JanusNC	DGMH	Best	Δ
email-Eu	Clust. coeff.	Node	0.49	0.52	0.31	JanusNC	0.03
		Edge	0.56	0.66	0.69	JanusNC	0.10
		Triangle	0.82	0.80	0.83	DGMH	0.01
		4-clique	0.89	0.86	0.91	DGMH	0.02
NDC-classes	Clust. coeff.	Node	0.61	0.66	0.64	DGMH	0.03
		Edge	0.71	0.89	0.92	JanusNC	0.18
		Triangle	0.80	0.95	0.97	JanusNC	0.15
		4-clique	0.74	0.95	0.97	JanusNC	0.21
email-Eu	Conn. comp.	Node	0.98	0.99	1.00	JanusNC	0.01
		Edge	0.70	0.84	0.77	DGMH	0.07
		Triangle	0.80	0.86	0.76	DGMH	0.04
		4-clique	0.41	0.54	0.42	DGMH	0.01
NDC-classes	Conn. comp.	Node	0.54	0.60	0.73	JanusNC	0.06
		Edge	0.58	0.62	0.79	JanusNC	0.04
		Triangle	0.26	0.29	0.23	Tie	0.03
		4-clique	0.19	0.20	0.20	Tie	0.01
email-Eu	Diameter	Node	2.82	2.82	1.67	JanusNC	0.00
		Edge	5.71	5.73	5.48	JanusNC	0.02
		Triangle	10.40	10.63	9.51	JanusNC	0.23
		4-clique	14.90	16.59	18.62	JanusNC	1.69
NDC-classes	Diameter	Node	4.45	4.50	4.91	JanusNC	0.05
		Edge	6.62	6.58	6.82	JanusNC	0.04
		Triangle	6.51	6.28	4.68	JanusNC	0.23
		4-clique	2.69	2.70	2.57	JanusNC	0.01
					Overall Best	**JanusNC**	

Table 51 reports a detailed comparison between JANUSNC and DGMH in reproducing structural properties of the email-Eu and NDC-classes datasets under the multilevel hypergraph decomposition adopted by DGMH [52]. We consider the node, edge, triangle, and 4-clique views induced by the k-level decomposition. For each configuration, the table reports the value measured on the real hypergraph, the values obtained by the generated hypergraphs, the method closest to the ground truth, and the corresponding absolute error.

Preprocessing and evaluation protocol. The DGMH values reported in the table are taken directly from the original paper, whereas JANUSNC is evaluated using the same preprocessing and evaluation

protocol. In particular, training is performed on the full dataset rather than restricting the input to its largest connected component. After applying the k-level decomposition, we remove hyperedges with cardinality larger than 25 for $k = 1$, and hyperedges with cardinality larger than 7 for $k \geq 2$, following the preprocessing adopted by DGMH and Do et al. [15].

The evaluation metrics follow the definitions used in the DGMH protocol. The *clustering coefficient* is computed for each k-level decomposition and then averaged across levels. The *connected-component metric* corresponds to the fraction of nodes contained in the largest connected component. Finally, the reported *diameter* is the 90% effective diameter, i.e., the minimum shortest-path distance required to cover 90% of connected node pairs.

We verify the consistency of this protocol by applying the same preprocessing and metric computation to the original datasets. The resulting real-data statistics reproduce the values reported by DGMH at each k level, providing a direct consistency check for the comparison. Therefore, differences between DGMH and JANUSNC are evaluated under matching preprocessing and metric definitions.

Results. Overall, JANUSNC provides the most consistent approximation to the real hypergraphs, obtaining the closest value in the majority of dataset, metric, and decomposition-level combinations. Its advantage is particularly evident for diameter and connectivity, for which it systematically remains close to the real hypergraphs across motif levels and both datasets.

For diameter, JANUSNC matches or nearly matches the real values in the node and edge views and generally maintains smaller errors at higher decomposition levels, where structural differences are typically amplified. This indicates that JANUSNC preserves not only local connectivity but also the characteristic path-length structure induced by higher-order interactions.

The comparison is more nuanced for the clustering coefficient. On email-Eu, DGMH is more accurate for some higher-order views, particularly the triangle and 4-clique decompositions, whereas JANUSNC is closer to the real hypergraph for lower-order structures. On NDC-classes, JANUSNC is more accurate across most decomposition levels, with particularly clear improvements for the edge, triangle, and 4-clique views.

A similar pattern emerges for the connected-component metric. JANUSNC shows stronger overall agreement with the real hypergraphs, especially on NDC-classes, where it clearly improves over DGMH in the node and edge views. On email-Eu, DGMH can be closer for selected higher-order views, but JANUSNC remains competitive and provides more consistent performance across decomposition levels. Ties occur only in a small number of cases, including some triangle- and 4-clique-level connectivity results on NDC-classes.

In summary, although DGMH is competitive and can outperform JANUSNC for specific higher-order clustering statistics, JANUSNC provides a more robust approximation across datasets, metrics, and k-level views. In particular, its consistent performance on connectivity and effective diameter suggests that it better preserves the structural organization of the original hypergraphs across multiple orders.

F.10 Sensitivity to β and sampling parameters

Since JANUS learns the distribution induced by sampled sub-hypergraphs, its performance may depend on both the β-VAE regularization strength and the sampling strategy used to construct the local training instances. We therefore evaluate the sensitivity of JANUS to the KL weight β, the biased random-walk parameters p and q, and the walk length. We vary one component at a time while keeping the remaining parameters fixed to the default configuration, $\beta = 10^{-6}$, $p = 2.0$, $q = 0.5$, and a walk length of 256. Table 52 reports a representative set of metrics spanning microscopic structure, mesoscopic motifs, global connectivity, and high-order dissimilarity; lower values are better in all cases.

Sensitivity to β. The default value $\beta = 10^{-6}$ provides the best overall performance. Increasing the regularization strength to $\beta = 10^{-3}$ leaves the global and high-order statistics relatively close to the default configuration: Hyper NetSimile changes from 0.898 to 0.962, Hyper Portrait Divergence from 0.116 to 0.118, and the line-graph edge error from $20{,}514$ to $20{,}594$. Fine-grained motif structure is more sensitive, however, with the open-triangle error increasing from 52.8 to 333.6. Reducing the regularization to $\beta = 10^{-9}$ results in a broader degradation, increasing Hyper NetSimile to 1.164, the node-degree distribution error to 4.661, and the line-graph edge error to $25{,}379$, while the triangle

error increases to 680.0. These results indicate that the selected β provides an effective balance between incidence reconstruction and latent regularization, while motif-level statistics are particularly sensitive to departures from this operating point.

Sensitivity to the walk bias. The sampling bias also affects the structural content of the sub-hypergraphs presented during training. The configuration used throughout the paper, $p = 2.0$ and $q = 0.5$, consistently obtains the lowest errors. An unbiased walk, $p = q = 1.0$, remains competitive on global statistics, with Hyper Portrait Divergence of 0.122 and a line-graph edge error of $20{,}663$, but increases the open-triangle error from 52.8 to 190.8. Reversing the bias to $p = 0.5$ and $q = 2.0$ produces a more pronounced degradation across all reported metrics, including Hyper NetSimile of 1.229, a line-graph edge error of $28{,}203$, and a triangle error of 557.4. This supports the use of the more explorative $p = 2.0$, $q = 0.5$ sampler, which exposes the model to a broader local context while retaining connected sub-hypergraph structure.

Sensitivity to the walk length. The walk length controls the amount of structural context contained in each sampled training instance. Reducing it from 256 to 128 substantially worsens all metrics: Hyper NetSimile increases from 0.898 to 1.814, the node-degree distribution error from 3.529 to 6.330, and the line-graph edge error from $20{,}514$ to $37{,}814$. The effect is particularly strong for local motifs, for which the open-triangle error increases from 52.8 to $5{,}566.8$. Increasing the walk length to 512 is considerably less detrimental, and global statistics remain relatively close to the default configuration; nevertheless, it does not improve performance and increases the triangle error to 801.2. Thus, a walk length of 256 provides the best trade-off between exposing sufficient structural context and preserving the local and mesoscopic patterns required for accurate generation.

Overall, the results in Table 52 show that JANUS is reasonably robust around its default configuration, particularly for global and high-order statistics. In contrast, fine-grained motif statistics are more sensitive to the sampling distribution, especially to the amount and type of context captured by the random walk. The default setting $\beta = 10^{-6}$, $p = 2.0$, $q = 0.5$, and walk length 256 achieves the lowest error across all reported metrics, supporting the configuration used in the main experiments.

Table 52: Sensitivity analysis over β, p–q, and random-walk length. Lower values are better for all evaluation metrics.

Sweep	p	q	β	Walk	NetSimile ↓	Portrait ↓	Degree ↓	LG Edges ↓	Triangles ↓
β	2.0	0.5	10^{-9}	256	1.164	0.130	4.661	25,379	680.0
β	2.0	0.5	10^{-6}	256	0.898	0.116	3.529	20,514	52.8
β	2.0	0.5	10^{-3}	256	0.962	0.118	3.899	20,594	333.6
p–q	0.5	2.0	10^{-6}	256	1.229	0.137	5.000	28,203	557.4
p–q	1.0	1.0	10^{-6}	256	1.041	0.122	3.853	20,663	190.8
p–q (paper)	2.0	0.5	10^{-6}	256	0.898	0.116	3.529	20,514	52.8
Walk	2.0	0.5	10^{-6}	128	1.814	0.170	6.330	37,814	5,566.8
Walk	2.0	0.5	10^{-6}	256	0.898	0.116	3.529	20,514	52.8
Walk	2.0	0.5	10^{-6}	512	1.004	0.121	4.065	21,875	801.2

G Ablation study details

G.1 Role of diffusion vs. clustering-based reconstruction.

Table 53 reports the performance obtained when the final node-identity assignment is carried out using only K-means on the decoded node representations after the local decoding stage. In this experiment, we disable the diffusion component to demonstrate that performance is not influenced by the post-processing strategy, but instead depends on the quality of generated representations. We evaluate clustering performance using three standard measures: Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), and Adjusted Mutual Information (AMI). ARI measures the agreement between the predicted clustering and the ground-truth node identities by comparing pairs of samples, while correcting for the agreement expected by chance. NMI quantifies the amount of shared information between the predicted and true partitions, normalized to account for the entropy of the two clusterings, and therefore ranges from no mutual dependence to perfect agreement. AMI is a chance-adjusted variant of mutual information, which corrects NMI for the similarity that can arise randomly, especially when the number or size of clusters differs across partitions. For all three

metrics, higher values indicate better clustering performance, with values close to one corresponding to near-perfect recovery of the ground-truth node identities.

The results show that this post-processing step is highly effective on the contact and email-Enron datasets: the ARI, NMI, and AMI values are close to one for contact-high-school, contact-primary-school, email-Enron, and email-Eu, indicating that the node decoder learns embeddings in which occurrences of the same underlying node are almost perfectly clustered. This supports the design choice of using decoded node representations as canonical descriptors for the entity-resolution step.

At the same time, the ablation highlights that node-assignment accuracy and structural generation quality are not always perfectly aligned. On NDC-classes, the ARI decreases substantially, while NMI and AMI remain relatively high. However, the structural evaluation does not show a substantial degradation in the quality of the generated hypergraph. This behavior is mainly due to the presence of isolated nodes in NDC-classes: assigning such nodes to different clusters can penalize clustering-based metrics, especially ARI, without necessarily altering the generated incidence structure or the higher-order organization of the network.

Table 53: Aggregated clustering metrics by dataset.

Dataset	ARI	NMI	AMI
contact-high-school	0.97774	0.99512	0.99490
contact-primary-school	0.99224	0.99816	0.99816
email-Enron	0.99788	0.99914	0.99912
email-Eu	0.99473	0.99972	0.99972
NDC-classes	0.50184	0.88380	0.86824

G.2 Dual-View VAE without the InfoNCE Loss

To further quantify this effect, we evaluate the quality of entity correspondence when explicit entity resolution is removed. The role of InfoNCE is to align representations corresponding to the same underlying node across overlapping sub-hypergraphs while separating representations associated with different entities. Without this contrastive signal, each view can still achieve low local reconstruction error, but there is no explicit objective enforcing identity consistency across views. Consequently, the model can reconstruct plausible local structures while failing to preserve the global correspondence required for a coherent hypergraph.

The results reported in Table 54 show that removing InfoNCE substantially degrades the structural quality of the generated hypergraphs on email-Enron. Across the five reported metrics, the InfoNCE-based model achieves an average improvement of 61.92% over the No-InfoNCE variant. The degradation is particularly pronounced for the number of open triangles, where InfoNCE yields a 99.44% improvement, while substantial gains are also observed for the node-degree distribution (54.54%) and the number of line-graph edges (53.90%). These results indicate that reconstruction loss alone is insufficient to produce a realistic joint latent representation.

G.3 Effect of cross-attention in paired diffusion.

Tables 55 and 56 report the ablation performed on the email-Enron and NDC-classes datasets, respectively, in which we remove the cross-attention layers from the paired diffusion model. In this variant, denoted JANUS*, the node and hyperedge denoisers no longer exchange information during the reverse denoising process. The results show that this coupling is an important component of the unconstrained model. On email-Enron, JANUS improves over JANUS* on almost all evaluation metrics, with an average relative improvement of 49.53%. A similar trend holds for NDC-classes, where the average relative improvement is 28.07%.

The largest gains appear on structural statistics that depend on the consistency between the generated node and hyperedge views. In particular, JANUS exactly matches the target number of nodes and the largest connected component diameter on email-Enron, while Janus* incurs non-zero errors on both metrics. JANUS also substantially reduces error on clique-expansion edges, closed triangles, open triangles, and hypergraph degree assortativity, with relative improvements of roughly 72% to 80%. On NDC-classes, we see similar gains for clique-expansion edges (86.23%), closed triangles (87.45%), and open triangles (98.96%), while node-count error decreases by 20.12%. These improvements

Table 54: Comparison of metrics with and without InfoNCE.

Metric	**InfoNCE**	**No InfoNCE**	**Δ (%)**
Hyper Net Simile	0.898	2.130	57.83
Hyper Portrait Div.	0.116	0.206	43.91
Node Degree Dist.	3.529	7.762	54.54
Line-graph edges	20514.4	44497.6	53.90
# Open triangles	52.8	9392.2	99.44
Avg. improvement			**61.92**

indicate that cross-attention helps the diffusion process generate node and hyperedge latents that are mutually compatible, leading to more coherent incidence patterns after decoding.

The same trend appears at the community and higher-order similarity levels. On email-Enron, hypergraph modularity improves by 64.63%, while Hyper Portrait Divergence and Hyper NetSimile improve by 39.83% and 44.95%, respectively. On NDC-classes, the corresponding improvements are smaller, with gains of 2.36%, 4.52%, and 22.63%, respectively. This confirms that the benefit of cross-attention is not limited to matching simple global statistics, but also extends to higher-order structural organization. The only metric where JANUS* performs slightly better on email-Enron is clique-expansion modularity, for which the difference is small compared with the gains observed across the remaining metrics. On NDC-classes, instead, clique-expansion modularity improves by 59.05%, while hypergraph degree assortativity is the only metric where removing cross-attention yields better results. Overall, the ablation supports the design choice of coupling the node and hyperedge diffusion streams: without cross-attention, the two latent processes can generate individually plausible representations, but they are less well aligned for reconstructing realistic hypergraph incidence structure.

Table 55: Comparison between JANUS with and without Cross-Attention (JANUS*). Improvement is reported as the relative improvement of JANUS over JANUS* on the email-Enron dataset.

	Janus	Janus*	Improvement (%)
Incidence Matrix Density	0.00209	0.00385	45.64492
Largest Connected Component Diameter	0.00000	0.40000	100.00000
Number of Nodes	0.00000	1.40000	100.00000
Number of Hyperedges	0.00000	0.00000	0.00000
Clique Expansion Number of Edges	13.20000	50.40000	73.80952
Line Graph Number of Edges	20514.40000	35809.60000	42.71257
Hypergraph Modularity	0.00472	0.01333	64.63398
Clique Expansion Modularity	0.00868	0.00789	-10.00372
Line Graph Modularity	0.02347	0.02837	17.30312
Bipartite Graph Modularity	0.02136	0.02845	24.93205
Normalized Mutual Information	0.09924	0.07788	27.41939
Hypergraph Degree Assortativity	0.02247	0.08182	72.53143
Number of Closed Triangles	115.20000	433.40000	73.41947
Number of Open Triangles	52.80000	258.00000	79.53488
Node Degree Distribution	3.52867	6.97394	49.40207
Hyperedge Size Distribution	0.30675	0.56283	45.49941
Hyper Portrait Divergence	0.11577	0.19240	39.82635
Hyper Net Simile	0.89838	1.63195	44.95066
Average Improvement			**49.53423**

G.4 Comparison with a VAE-only baseline

To directly assess whether latent diffusion is necessary for generation, we compare JANUS against a VAE-only baseline. The baseline retains the same encoder, incidence decoder, and, in the unconstrained setting, entity-resolution mechanism, but removes the latent diffusion component. We set $\beta = 1$ at training time, and at generation time we sample node and hyperedge latent representations directly from an isotropic Gaussian prior,

$$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),$$

Table 56: Comparison between JANUS with and without Cross-Attention (JANUS*). Improvement is reported as the relative improvement of JANUS over JANUS* on the NDC-classes dataset.

	Janus	Janus*	Improvement (%)
Incidence Matrix Density	0.00460	0.00750	38.61000
Largest Connected Component Diameter	0.80000	1.00000	20.00000
Number of Nodes	442.20000	553.60000	20.12000
Number of Hyperedges	0.00000	0.00000	0.00000
Clique Expansion Number of Edges	159.00000	1155.00000	86.23000
Line Graph Number of Edges	82433.80000	86629.40000	4.84000
Hypergraph Modularity	0.12800	0.13100	2.36000
Clique Expansion Modularity	0.07100	0.17500	59.05000
Line Graph Modularity	0.11000	0.13600	19.28000
Bipartite Graph Modularity	0.11500	0.14200	19.02000
Normalized Mutual Information	0.24300	0.21500	12.96000
Hypergraph Degree Assortativity	0.16700	0.10800	-53.83000
Number of Closed Triangles	1638.80000	13058.60000	87.45000
Number of Open Triangles	1229.20000	118427.40000	98.96000
Node Degree Distribution	5.01500	8.21400	38.95000
Hyperedge Size Distribution	1.52500	2.00900	24.08000
Hyper Portrait Divergence	0.42100	0.44100	4.52000
Hyper Net Simile	6.04100	7.80800	22.63000
Average Improvement			**28.07000**

corresponding to $\mu = 0$ and $\sigma = 1$. The resulting latents are then passed directly to the pretrained decoder, and the remainder of the generation pipeline is kept unchanged. This comparison therefore isolates the contribution of learning the latent generative distribution through diffusion: it tests whether accurate reconstruction together with direct sampling from a Gaussian latent prior is sufficient to generate realistic hypergraphs.

Table 57 shows that this is not the case. To provide a compact comparison while covering the structural scales considered in our evaluation, we report Hyper NetSimile and Hyperedge Portrait Divergence as high-order dissimilarity measures, node-degree distribution as a microscopic statistic, open-triangle error as a mesoscopic statistic, and line-graph edge error as a macroscopic statistic. In the node-set constrained setting, we additionally report the number of recovered hyperedges and the hyperedge Jaccard coefficient. These metrics emphasize the categories where the strongest methods in the original evaluation are most clearly separated.

Unconstrained generation. The VAE-only baseline performs substantially worse than JANUS across all reported structural scales. The largest difference occurs for mesoscopic structure: the open-triangle error decreases from $176{,}836 \pm 812$ with direct prior sampling to only 52.8 ± 32.5 with JANUS, corresponding to a 99.97% reduction. Microscopic structure exhibits a similarly clear improvement, with the node-degree distribution error decreasing from 18.722 ± 0.539 to 3.529 ± 0.367 (81.15%). At the macroscopic scale, JANUS reduces the line-graph edge error from $32{,}093 \pm 3{,}421$ to $20{,}514 \pm 3{,}064$ (36.08%).

The same trend is observed for higher-order similarity. Hyper NetSimile decreases from 7.963 ± 0.151 to 0.898 ± 0.068, an 88.72% reduction, while Hyperedge Portrait Divergence decreases from 0.757 ± 0.018 to 0.116 ± 0.009, corresponding to an 84.70% improvement. Thus, although the VAE decoder is trained to reconstruct observed incidence structures, samples drawn directly from the Gaussian prior do not reproduce the latent configurations required to preserve the structural statistics of the original hypergraph.

Node-set constrained generation. The advantage of latent diffusion is even more pronounced when the node set is fixed. Hyper NetSimile decreases from 6.833 ± 0.153 for the VAE-only model to 0.139 ± 0.018 for JANUS, while Hyperedge Portrait Divergence decreases from 0.638 ± 0.023 to 0.046 ± 0.003, corresponding to improvements of 97.96% and 92.78%, respectively. The node-degree distribution error is reduced by 94.47%, from 15.341 ± 0.715 to 0.848 ± 0.110, and the line-graph edge error decreases from $28{,}347 \pm 2{,}577$ to $2{,}228 \pm 822$ (92.14%). As in the unconstrained setting,

motif-level structure is particularly sensitive: the open-triangle error falls from $164{,}902 \pm 2{,}401$ to 81.8 ± 24.9, a 99.95% reduction.

The difference is also evident in direct reconstruction. JANUS recovers $1{,}450.2 \pm 4.9$ hyperedges compared with only 91.4 ± 5.5 for the VAE-only baseline, corresponding to a $15.87\times$ increase. Similarly, the hyperedge Jaccard coefficient rises from 0.0312 ± 0.0020 to 0.9215 ± 0.0060, a $29.56\times$ improvement. These results show that fixing the node set alone does not make direct prior sampling sufficient: realistic hyperedge structure still requires samples that follow the latent distribution learned from observed sub-hypergraphs.

Overall, the comparison in Table 57 demonstrates that the strong reconstruction capability of the dual-view autoencoder is not sufficient for generation. The decoder can accurately map latent representations associated with observed samples back to incidence structures, but arbitrary samples from the standard Gaussian prior do not reliably lie in regions corresponding to realistic hypergraphs. The latent diffusion model is therefore not merely an auxiliary component: it learns how to generate plausible node and hyperedge latent configurations before decoding. This role is particularly important for mesoscopic motifs and higher-order structure, where direct prior sampling produces the largest errors.

Table 57: Comparison between VAE-only generation and JANUS in the size-conditioned and node-set-constrained settings. Arrows indicate whether lower (↓) or higher (↑) values are better.

Unsconstrained generation			
Metric	VAE-only	Janus	Δ
Hyper NetSimile ↓	7.963 ± 0.151	0.898 ± 0.068	88.72% ↓
Hyperedge Portrait Div. ↓	0.757 ± 0.018	0.116 ± 0.009	84.70% ↓
Node Degree Dist. ↓	18.722 ± 0.539	3.529 ± 0.367	81.15% ↓
Line-graph edges err. ↓	$32{,}093 \pm 3{,}421$	$20{,}514 \pm 3{,}064$	36.08% ↓
Open triangles err. ↓	$176{,}836 \pm 812$	52.8 ± 32.5	99.97% ↓
Constrained generation			
Metric	VAE-only	Janus	Δ
Hyper NetSimile ↓	6.833 ± 0.153	0.139 ± 0.018	97.96% ↓
Hyperedge Portrait Div. ↓	0.638 ± 0.023	0.046 ± 0.003	92.78% ↓
Node Degree Dist. ↓	15.341 ± 0.715	0.848 ± 0.110	94.47% ↓
Line-graph edges err. ↓	$28{,}347 \pm 2{,}577$	$2{,}228 \pm 822$	92.14% ↓
Open triangles err. ↓	$164{,}902 \pm 2{,}401$	81.8 ± 24.9	99.95% ↓
Recovered hyperedges ↑	91.4 ± 5.5	$1{,}450.2 \pm 4.9$	$15.87\times$ ↑
Hyperedge Jaccard ↑	0.0312 ± 0.0020	0.9215 ± 0.0060	$29.56\times$ ↑

H Performance analysis experiment

Table 58: Training and sampling wall-clock times on the email-Enron dataset for both the unconstrained and node-set constrained settings.

Model	Training Time	Sampling Time
JANUS-β-VAE	1h 34m	7s
JANUS-LDM	10h 04m	-
JANUS	11h 38m	42s
JANUSNC-β-VAE	1h 18m	25s
JANUSNC-LDM	9h 25m	-
JANUSNC	10h 43m	9m 33s

Table 58 reports the computational cost of the proposed models on email-Enron, averaged over 5 experiments and measured on a desktop machine equipped with 24 CPUs, 188 GB of RAM, and an NVIDIA RTX 5090 GPU with 32 GB of memory.

Training. The β-VAE components are relatively inexpensive: JANUS-β-VAE trains in 1h 34m, while JANUSNC-β-VAE trains in 1h 18m. The latent diffusion models require a higher training cost, reaching 10h 04m for JANUS-LDM and 9h 25m for JANUSNC-LDM. This overhead is expected,

since the full models train diffusion denoisers in the learned latent space rather than relying only on direct decoding from the variational autoencoder.

Sampling. At generation time, the decoding stage itself introduces only a modest overhead: JANUS-β-VAE produces a decoded sample in 7s, whereas JANUSNC-β-VAE requires 25s. When considering the full generation pipeline, which includes latent sampling and decoding, the unconstrained JANUS remains fast, requiring 42s to generate a hypergraph. By contrast, JANUSNC requires 9m 33s. This gap shows that, although the node-set constrained setting is slightly cheaper to train, it shifts most of the computational burden to sampling, where reconstructing interactions over a fixed node set is substantially more expensive.

H.1 Computational complexity

To analyze the complexity characterization of JANUS, let M be the number of hyperedges in the original hypergraph, n and m the typical numbers of nodes and hyperedges in a sampled sub-hypergraph, S the number of sub-hypergraphs generated to reach the target size, and T the number of reverse diffusion steps. Since JANUS constructs one training sample per original hyperedge, the computational cost per training epoch is approximately $O(M(n+m)^2)$. This expression captures the dominant local representation-learning and attention operations, while omitting fixed architectural dimensions and optimization constants.

At inference time, generating S sub-hypergraphs through T reverse diffusion steps costs $O(ST(n+m)^2)$. For unconstrained JANUS, entity resolution additionally clusters approximately Sn generated node occurrences into N global entities, resulting in an approximate cost of $O(SnN)$ when the number of clustering iterations is treated as constant. The resulting end-to-end complexity is therefore

$$O((M+ST)(n+m)^2 + SnN).$$

For JANUSNC, where node identities are prescribed, and entity resolution is unnecessary, the last term is omitted. Importantly, the quadratic dependence concerns the size n and m of the local sub-hypergraphs rather than the complete hypergraph. The overall cost nevertheless grows with the number of training and generated sub-hypergraphs.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction accurately reflect the papers contributions and scope: they describe JANUS as a dual-view latent diffusion framework that jointly models nodes and hyperedges through a β-VAE and paired latent diffusion with cross-view conditioning, matching the method in Sec. 4 and Secs. 4.1–4.2 . They also state the single-hypergraph, unconstrained and node-set-constrained generation settings, which are formalized in Sec. 3.1 . The claimed evaluation framework is supported by Sec. 5, which covers micro-, meso-, macro-, modularity, higher-order dissimilarity, and reconstruction metrics, with implementation details in App. E and full results in App. F . Finally, the experimental scope matches the introductions claims: the paper evaluates five real-world datasets and nine baselines, with additional comparisons to DGMH and ablations in Secs. 5.2–5.3

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper explicitly discusses limitations in the Limitations and Future Work paragraph of Sec. 6. It notes that JANUS relies on a sampling operator to construct sub-hypergraphs and that the current work mainly explores biased random-walk sampling. It also identifies future directions for the unconstrained setting, including a more systematic study of sampling strategies and the integration of classifier-free guidance to control generation across structural scales

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.

- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not present formal theoretical results such as theorems, propositions, or lemmas requiring stated assumptions and complete proofs. Its contributions are methodological and empirical, focusing on the JANUS framework, evaluation protocol, experiments, comparison with baselines, and ablation studies, as described in Secs. 4–5 and Apps. B–G

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper provides the information needed to reproduce the main experimental results. Sec. 5 defines the experimental setup, evaluation framework, baselines, datasets, and main comparison protocol, including the five real-world datasets and nine baselines . The appendix further reports implementation and reproducibility details, including sub-hypergraph sampling, train/validation split, batch size, `node2vec` feature extraction, β-VAE and diffusion training protocols, checkpoint selection, EMA, and diffusion schedule in App. B . App. C specifies model hyperparameters such as latent dimensionality, learning rates, β value, weight decay, diffusion steps, and sampler parameters . Dataset preprocessing and summary details are reported in App. D, and the paper provides the code and datasets at `https://github.com/hypernetwork-research-group/janus`

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may

be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The paper states that both code and datasets are publicly available at `https://github.com/hypernetwork-research-group/janus` to support reproducibility. The experimental protocol is described in Sec. 5, including datasets, baselines, and evaluation metrics, while the appendix provides implementation details, hyperparameters, sampling strategy, training/validation split, and training procedures needed to reproduce the main results

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: The paper specifies the main experimental settings in Sec. 5, including the evaluation protocol, datasets, baselines, and checkpoint-selection strategy based on generated-hypergraph metrics rather than diffusion loss. App. B and App. C provide the training and implementation details, including the 90/10 train-validation split, batch size of 32, biased random-walk sampler parameters, `node2vec` feature extraction settings, optimizer choice, learning rates, weight decay, EMA decay, latent dimension, β-VAE training length, diffusion schedule with $T = 1000$ steps, and checkpoint criteria.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The paper reports that results are averaged over 5 generated hypergraphs and provides the full numerical values in App. F. Each value is reported as the mean and standard deviation over multiple (5) runs, thereby indicating statistical variability.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: The paper reports the compute resources used for the experiments, specifying a desktop machine with 24 CPUs, 188 GB of RAM, and an NVIDIA RTX 5090 GPU with 32 GB of memory. It also reports the average training and generation times in App. H, providing the main hardware and execution-time information needed to reproduce the experiments.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research focuses on algorithmic development for hypergraph generation and evaluation on publicly available real-world hypergraph datasets. It does not involve human subjects, private/sensitive personal data collection, or obviously harmful applications. The paper also provides code and datasets to support transparency and reproducibility, and we assume it conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses the broader impact of the proposed hypergraph generation framework in the conclusion. On the positive side, JANUS can support the study of higher-order relational systems when real interaction data are incomplete, difficult to collect, or sensitive to share, by enabling simulation, hypothesis testing, and synthetic data generation. On the negative side, the paper notes that generated hypergraphs from sensitive domains may still pose privacy or misuse risks if they reveal structural patterns of the original data or are interpreted as real observations without appropriate safeguards.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper proposes a hypergraph generation framework. Neither the model nor the benchmark hypergraph datasets used appear to pose a high risk for misuse necessitating specific release safeguards beyond standard open-source practices.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [No]

    Justification: The paper properly credits the creators of existing assets, including the baseline methods, related hypergraph generators, evaluation metrics, and datasets used in the experiments. For example, it cites the benchmark datasets from Benson et al., the baseline methods such as HYPERPA, SUBSETSAMPLING, NAIVEPA, THERA, HYPERPLR, HYPERDK, and HYPERLAP/HYPERLAP+, as well as external tools and methods such as NODE2VEC and higher-order dissimilarity measures. However, the specific licenses and terms of use for these datasets, codebases, and external methods are not explicitly reported in the paper or appendix.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The primary new asset is the source code for the proposed JANUS framework, which is made publicly available. The repository includes the required code environment and commands needed to reproduce the experiments and execute the JANUS model, so the introduced assets are documented alongside their release.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The research does not involve crowdsourcing experiments or studies with human participants.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The research does not involve human participants; therefore, IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: Large language models were used only for grammar checking and minor language polishing.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.