AD-DHGNN: ADAPTIVE DUAL-CONTROLLED DYNAMIC HYPERGRAPH NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Dynamic hypergraph neural networks adapt high-order relations to evolving node representations, yet most existing constructions rely on fixed hyperedge counts or hyperedge cardinality constraints. Such fixed structural choices can produce redundant, oversized, or fragmented hypergraphs when the required relation structure varies across layers and datasets. We propose the Adaptive Dual-Controlled Dynamic Hypergraph Neural Network (AD-DHGNN), which treats global hyperedge count and local hyperedge cardinality as two complementary structural variables. AD-DHGNN first generates posterior-weighted candidate hyperedges from anchor-based node similarities, avoiding dense node-pair affinity estimation. It then applies a cardinality controller that assigns data-dependent capacity budgets to individual candidates, followed by a count controller that selects a compact and representative subset of hyperedges. The resulting sparse hypergraph is used for gated dynamic message passing. Experiments on seven node-classification benchmarks show that AD-DHGNN improves predictive performance over the evaluated baselines while constructing compact and low-redundancy dynamic hypergraphs. These results demonstrate that explicitly determining hyperedge count and hyperedge cardinality is a central principle for scalable dynamic hypergraph reconstruction.

1 INTRODUCTION

Graphs represent relational data, but ordinary edges encode only pairwise interactions Kipf & Welling (2017); Veličković et al. (2018). Hypergraphs extend this representation by connecting groups of nodes within a single hyperedge, allowing hypergraph neural networks to model and propagate high-order relations Feng et al. (2019).

Many hypergraph neural networks rely on a predefined incidence structure Yadati et al. (2019); Gao et al. (2022). Their effectiveness therefore depends on whether the supplied hyperedges reflect relations useful for the task. Dynamic methods reconstruct hyperedges from learned node representations Jiang et al. (2019); Zhou et al. (2023), allowing the structure to change across layers. Reconstruction introduces two decisions: how many hyperedges to retain and how many nodes each hyperedge should contain. These decisions are complementary. Retaining fewer hyperedges does not prevent an individual hyperedge from expanding across unrelated nodes, while limiting cardinality does not remove numerous overlapping candidates. Aggressive control on either axis may also discard useful relations. Prior dynamic methods can adapt hyperedge membership or count Jiang et al. (2019); Zhou et al. (2023), but adapting one property does not automatically regulate the other. This motivates separate, coordinated control of local hyperedge cardinality and global hyperedge count.

We propose the Adaptive Dual-Controlled Dynamic Hypergraph Neural Network (AD-DHGNN). It selects fixed degree-guided anchors and uses an Approximate Personalized Propagation of Neural Predictions (APPNP) encoder Klicpera et al. (2019) to obtain graph-aware node representations. At each dynamic layer, node–anchor similarities define Voronoi regions and are converted by anchor-specific two-component Gaussian mixtures into posterior-weighted candidate hyperedges. Each Voronoi region provides a cardinality bound for its candidate; when this bound is exceeded, nodes with the strongest posterior evidence are retained. Candidates with identical supports are then deduplicated. For count control, a spectral criterion estimates the number of dominant incidence patterns,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

while a coverage-aware greedy procedure orders candidates by their contribution to node and region coverage. The spectral and coverage criteria jointly determine how many candidates to retain. The resulting weighted incidence matrix is used for gated hypergraph message passing. Computing node–anchor similarities requires $N \times K$ comparisons rather than a dense $N \times N$ node-pair affinity matrix, where K is the number of anchors. Across seven node-classification benchmarks, AD-DHGNN achieves the highest reported mean among the available baselines on each dataset. Ablation and structural analyses examine how cardinality and count control affect hyperedge size, redundancy, and predictive performance.

Our contributions are summarized as follows:

- We propose AD-DHGNN, an adaptive dual-controlled dynamic hypergraph neural network that jointly models and regulates two complementary structural variables, namely local hyperedge cardinality and global hyperedge count.
- We introduce an adaptive hyperedge cardinality controller that uses anchor-induced Voronoi regions to estimate candidate-specific capacity bounds. Posterior evidence is then used to retain the most strongly supported nodes when a candidate exceeds its estimated capacity.
- We develop an adaptive hyperedge count controller that combines spectral structure with coverage-aware greedy selection. It estimates how many representative hyperedges should be retained and removes globally redundant candidates while preserving complementary structural information.

2 Related Work

2.1 Traditional and Structure-Learning Hypergraph Neural Networks

Traditional hypergraph neural networks typically perform message passing over a predefined hypergraph structure. Representative spectral approaches, such as HGNN, HGNN+, and HyperGCN, extend graph convolution to model higher-order relations among multiple nodes Feng et al. (2019); Gao et al. (2022); Yadati et al. (2019). HNHN, UniGNN, AllSet, and ED-HNN further develop more flexible node–hyperedge message-passing mechanisms Dong et al. (2020); Huang & Yang (2021); Chien et al. (2022); Wang et al. (2023). Recent studies also enhance hypergraph representation learning from richer structural perspectives, for example through cellular sheaves or frequency-aware modeling Duta et al. (2023); Huang et al. (2025).

Beyond improving propagation operators, another line of research focuses on learning or refining the hypergraph structure itself. Dynamic Hypergraph Structure Learning jointly optimizes hypergraph structure and predictive objectives Zhang et al. (2018), while Hypergraph Structure Learning (HSL) refines a given incidence structure through differentiable sampling and contrastive learning Cai et al. (2022). These studies demonstrate the benefit of adapting hypergraph topology to the learning objective, but they mainly focus on learning or refining an existing hypergraph rather than reconstructing a compact structure at each network layer.

2.2 Dynamic Hypergraph Neural Networks

Dynamic hypergraph neural networks further extend structure learning by reconstructing hyperedges from evolving node representations. DHGNN dynamically generates higher-order relations through neighborhood search or clustering, whereas DHKH emphasizes structurally important key hyperedges Jiang et al. (2019); Kang et al. (2022). TDHNN further adapts hyperedge features, node memberships, and hyperedge count, while DHHNN combines dynamic hypergraph construction with hyperbolic representation learning Zhou et al. (2023); Mei et al. (2025). More recently, FDHNN introduces a fission-based reconstruction mechanism with collaborative hypergraph generation Xu et al. (2026), further illustrating the importance of adapting hypergraph structures during representation learning.

Although these methods improve dynamic hypergraph construction from different perspectives, they do not explicitly formulate the local size of individual hyperedges and the global number of retained

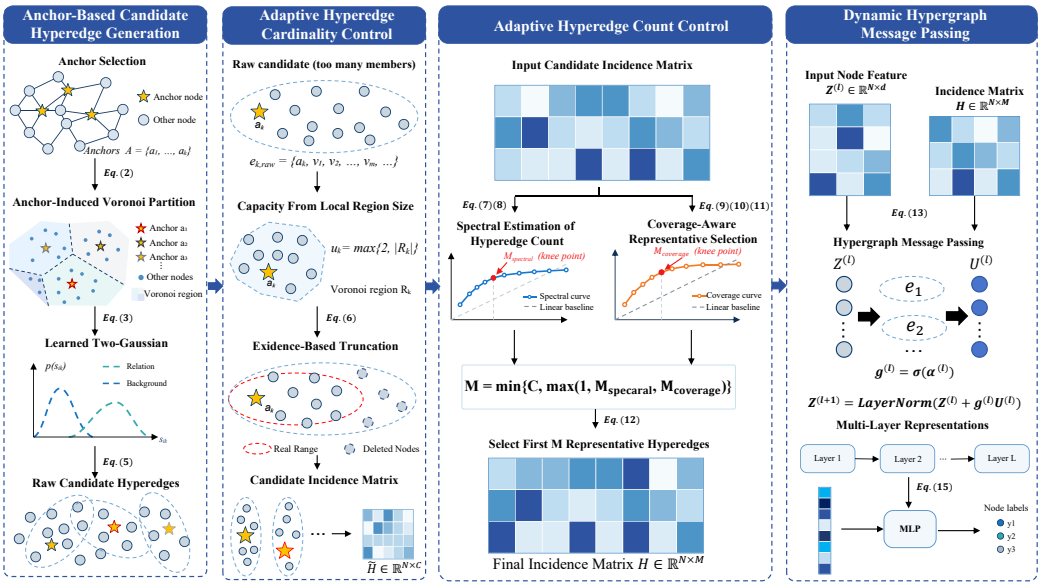

Figure 1: Overview of AD-DHGNN. From left to right, the four modules construct candidate hyperedges, control their cardinality, select a representative subset, and perform dynamic hypergraph message passing for node classification.

hyperedges as two coordinated structural variables. In contrast, AD-DHGNN decomposes layer-wise dynamic reconstruction into these two complementary decisions. It regulates local hyperedge cardinality using candidate-specific capacity constraints and controls the global structural budget through spectral redundancy and coverage-aware subset selection. This dual-control design allows the model to reduce both oversized local relations and globally redundant hyperedges.

3 METHOD

To address the redundant and oversized hyperedges produced during dynamic hypergraph reconstruction, we propose the Adaptive Dual-Controlled Dynamic Hypergraph Neural Network (AD-DHGNN), which adaptively regulates both hyperedge length and quantity. An overview of AD-DHGNN is provided in Figure 1.

3.1 ANCHOR-BASED CANDIDATE HYPEREDGE GENERATION

Degree-Guided Anchor Selection Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}_g)$ denote an attributed graph with N nodes, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the node set, $\mathbf{X} \in \mathbb{R}^{N \times d_0}$ is the node feature matrix, and $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix. We measure the structural importance of node v_i by its degree $d_i = \sum_{j=1}^{N} \mathbf{A}_{ij}$, where $\mathbf{A}_{ij}$ denotes the adjacency between nodes v_i and v_j. We rank all nodes according to their degree values and select the K nodes with the largest degrees as the fixed anchor set $\mathcal{A} = \{a_1, \ldots, a_K\}$, where K is a dataset-specific hyperparameter. High-degree nodes tend to occupy structurally central positions and therefore provide stable reference points for subsequent candidate hyperedge generation.

After selecting the anchors, we apply an Approximate Personalized Propagation of Neural Predictions (APPNP) encoder to the input graph, obtaining the graph-aware node representation $\mathbf{Z}_g = f_{\mathrm{E}}(\mathbf{X}, \mathbf{A}) \in \mathbb{R}^{N \times d}$, where f_{E} denotes the APPNP encoder and d is the hidden dimension. The representation $\mathbf{Z}_g$ initializes the dynamic hypergraph network and is subsequently used to compute the similarities between each anchor and all nodes.

Anchor-Induced Voronoi Partition. At each dynamic layer, AD-DHGNN reconstructs candidate hyperedges from the current node representations $\mathbf{Z} \in \mathbb{R}^{N \times d}$, where $\mathbf{Z} = \mathbf{Z}_g$ at the first layer. We

combine the stable graph representation with the current node state as

$$\mathbf{Q} = \mathrm{RowNorm}\left(\mathrm{RowNorm}(\mathbf{Z}_g) + \mathrm{RowNorm}(\mathbf{Z})\right), \tag{1}$$

where $\mathrm{RowNorm}(\cdot)$ normalizes each row to unit ℓ_2 norm. This fusion preserves information from the input graph while allowing the reconstructed hypergraph to evolve with the updated node representations.

Let $\mathbf{q}_i$ denote the representation of node v_i in $\mathbf{Q}$. For anchor a_k, its cosine similarity with node v_i is $s_{ik} = \cos(\mathbf{q}_i, \mathbf{q}_{a_k})$. Each node is assigned to its most similar anchor, yielding

$$c_i = \arg\max_{1 \le k \le K} s_{ik}, \qquad \mathcal{R}_k = \{v_i \in \mathcal{V} \mid c_i = k\}. \tag{2}$$

Here, c_i denotes the anchor assigned to node v_i, and $\mathcal{R}_k$ is the Voronoi region associated with anchor a_k. Each anchor is explicitly assigned to its own region, while other similarity ties are resolved according to the fixed anchor order. Since the node representations evolve across dynamic layers, the Voronoi partition is recomputed at every layer. The region size $|\mathcal{R}_k|$ characterizes the local structural scale around a_k and is subsequently used by the adaptive hyperedge cardinality controller.

Posterior Probability Estimation. Since $|\mathcal{R}_k|$ only constrains the candidate capacity without specifying node–anchor relation membership, candidate membership is inferred independently for each anchor from its node–anchor similarity distribution. For anchor a_k, we fit a two-component Gaussian mixture model (GMM) to its $N-1$ similarities with all other nodes.

The mixture weights, together with the corresponding means and variances, are estimated by expectation maximization (EM) Dempster et al. (1977). In the E-step, $\gamma_{ik,c}$ denotes the posterior responsibility of component $c \in \{-,+\}$ for similarity s_{ik}, which is obtained from the current mixture weight and Gaussian likelihood of that component. The M-step aggregates these soft assignments and computes the mixture weight as

$$\omega_{k,c} = \frac{\sum_{i \neq a_k} \gamma_{ik,c}}{\sum_{c' \in \{-,+\}} \sum_{i \neq a_k} \gamma_{ik,c'}} = \frac{1}{N-1} \sum_{i \neq a_k} \gamma_{ik,c}. \tag{3}$$

Thus, $\omega_{k,c}$ represents the proportion of the soft assignments explained by component c, with $\omega_{k,+} + \omega_{k,-} = 1$. After EM convergence, the two components are ordered such that $\mu_{k,+} \geq \mu_{k,-}$, where the higher-mean component is treated as the relation-supported component and the lower-mean component as the background component (Appendices A.1 and A.4).

Given the observed similarity s_{ik}, the posterior probability that node v_i belongs to the relation-supported component of anchor a_k is

$$p_{ik} = \frac{\omega_{k,+}\mathcal{N}(s_{ik}; \mu_{k,+}, \sigma^2_{k,+})}{\omega_{k,+}\mathcal{N}(s_{ik}; \mu_{k,+}, \sigma^2_{k,+}) + \omega_{k,-}\mathcal{N}(s_{ik}; \mu_{k,-}, \sigma^2_{k,-})}. \tag{4}$$

Here, $\mathcal{N}(s; \mu, \sigma^2)$ denotes the Gaussian density. The posterior $p_{ik} \in [0,1]$ measures the strength of the relation between node v_i and anchor a_k, and a larger value indicates stronger evidence that v_i should participate in the candidate generated by a_k.

Candidate Hyperedge Generation. The posterior probability p_{ik} only quantifies the relation strength between node v_i and anchor a_k, and we further use it to construct candidate hyperedges. A non-anchor node is included when $p_{ik} > 1/2$, corresponding to the maximum-a-posteriori decision between the relation-supported and background components (Appendix A.2), while the source anchor itself is always retained. The raw candidate generated by anchor a_k is

$$e_{k,\mathrm{raw}} = \{a_k\} \cup \left\{ v_i \in \mathcal{V} \setminus \{a_k\} \mid p_{ik} > \frac{1}{2} \right\}. \tag{5}$$

The source anchor receives an incidence weight of one, whereas each selected non-anchor node receives its posterior probability p_{ik}. Candidates containing only their source anchors are discarded. For each valid raw candidate, we further define the posterior log-odds evidence as $\xi_{ik} = \log[p_{ik}/(1-p_{ik})]$, which measures the strength of the relation-supported membership of node v_i with respect to anchor a_k. The resulting raw candidates $e_{k,\mathrm{raw}}$, together with their posterior membership probabilities p_{ik} and posterior evidence ξ_{ik}. Accordingly, the resulting raw candidate hyperedges $\mathcal{E} = \{e_{k,\mathrm{raw}}\}_{k=1}^{K}$ are subsequently passed to the adaptive hyperedge cardinality controller.

3.2 Adaptive Hyperedge Cardinality Control

Adaptive Estimation and Regulation of Hyperedge Cardinality. To prevent local over-expansion of candidate hyperedges, we introduce an adaptive cardinality controller. For each anchor a_k, Section 3.1 produces raw candidate hyperedges $\{e_{k,\mathrm{raw}}\}$, together with its posterior membership probability p_{ik} and posterior log-odds evidence ξ_{ik}.

We use the anchor-induced Voronoi region $\mathcal{R}_k$ obtained during candidate generation to characterize the local structural scale around a_k, and define the candidate-specific length upper bound as $u_k = \max\{2, |\mathcal{R}_k|\}$ (Appendix A.3). If $e_{k,\mathrm{raw}}$ exceeds its adaptive upper bound, we preserve the source anchor and retain the non-anchor members with the strongest posterior log-odds evidence:

$$e_k = \begin{cases} e_{k,\mathrm{raw}}, & |e_{k,\mathrm{raw}}| \le u_k, \\ \{a_k\} \cup \mathrm{TopNodes}_{u_k-1}\left(e_{k,\mathrm{raw}} \setminus \{a_k\}; \xi_{ik}\right), & |e_{k,\mathrm{raw}}| > u_k. \end{cases} \tag{6}$$

Here, $\mathrm{TopNodes}_{u_k-1}$ selects the $u_k - 1$ non-anchor members with the largest posterior log-odds evidence. For each non-singleton candidate, this operation ensures $2 \le |e_k| \le u_k$; when truncation is required, it maximizes retained posterior log-odds evidence among feasible subsets of size u_k (Appendix A.5).

Exact Candidate Deduplication. After length regulation, candidates with identical node supports are deduplicated. If multiple candidates share the same support, we retain the one with the larger aggregate posterior evidence. Candidates that only partially overlap are not removed at this stage, since their global redundancy is handled by the subsequent count controller.

After deduplication, the remaining candidates are re-indexed as $\widetilde{\mathcal{E}} = \{\widetilde{e}_k\}_{k=1}^{C}$. Let $\widetilde{a}_k$ denote the source anchor of candidate $\widetilde{e}_k$, and let $\widetilde{p}_{ik}$ denote the posterior membership probability inherited from its source anchor. These candidates form the weighted candidate incidence matrix $\widetilde{\mathbf{H}} \in \mathbb{R}^{N\times C}$, where the source anchor $\widetilde{a}_k$ receives weight 1, each retained non-anchor member v_i receives weight $\widetilde{p}_{ik}$, and all other entries are zero. The resulting $\widetilde{\mathbf{H}}$ is then passed to the adaptive hyperedge count controller for global redundancy reduction.

3.3 Adaptive Hyperedge Count Control

Spectral Estimation of Hyperedge Count. To alleviate global structural redundancy caused by excessive and overlapping candidate hyperedges, we introduce an adaptive hyperedge count controller. The candidate incidence matrix $\widetilde{\mathbf{H}} \in \mathbb{R}^{N\times C}$ obtained in Section 3.2 contains C valid candidate hyperedges. We use the spectral structure of $\widetilde{\mathbf{H}}$ to determine how many hyperedges to retain, then select a representative subset.

We first construct the normalized Gram matrix $\mathbf{G} \in \mathbb{R}^{C\times C}$ and denote its eigenvalues by $\lambda_1, \ldots, \lambda_C$:

$$\mathbf{G}_{cj} = \frac{\widetilde{\mathbf{H}}_{:c}^{\top}\widetilde{\mathbf{H}}_{:j}}{\left\|\widetilde{\mathbf{H}}_{:c}\right\|_2 \left\|\widetilde{\mathbf{H}}_{:j}\right\|_2}, \qquad \mathrm{spec}(\mathbf{G}) = \{\lambda_r\}_{r=1}^{C}. \tag{7}$$

Here, $\mathbf{G}_{cj}$ is the cosine similarity between the weighted incidence patterns of candidates c and j, and $\mathrm{spec}(\mathbf{G})$ is the eigenvalue set of $\mathbf{G}$. Since $\mathbf{G}$ is positive semidefinite, $\lambda_r \ge 0$ for all r (Appendix A.7). We then sort the eigenvalues in descending order as $\lambda_{(1)} \ge \cdots \ge \lambda_{(C)} \ge 0$ and define the cumulative spectral contribution of the first m largest eigenvalues as $S_m = \sum_{r=1}^{m} \lambda_{(r)} / \sum_{r=1}^{C} \lambda_{(r)}$. As m increases, S_m reflects how rapidly the dominant structural information is accumulated. We therefore determine the spectral count by locating the point at which the cumulative spectral curve deviates most from the linear baseline $\frac{m}{C}$ Satopaa et al. (2011):

$$M_{\mathrm{spectral}} = \arg\max_{1\le m\le C} \left(S_m - \frac{m}{C}\right). \tag{8}$$

Here, M_{spectral} estimates the number of dominant structural directions represented by the candidate hyperedges. When multiple positions attain nearly identical maximum scores, the larger index is selected to avoid overly aggressive pruning. If no meaningful knee is detected, all candidate hyperedges are conservatively retained.

Coverage-Aware Representative Hyperedge Selection. The spectral count determines how many dominant structural directions should be preserved, but it does not specify which candidate hyperedges should be retained. We therefore construct a candidate ordering by jointly considering node-level soft coverage and anchor-induced Voronoi-region coverage.

Suppose that we have a selected candidate hyperedge set $\mathcal{S}$, and denote $f_i(\mathcal{S}) = \max_{e_j \in \mathcal{S}} \widetilde{\mathbf{H}}_{ij}$ as the current soft coverage of node v_i. The global node-level soft coverage is then defined as

$$F_{\text{node}}(\mathcal{S}) = \frac{\sum_{i=1}^{N} f_i(\mathcal{S})}{\sum_{i=1}^{N} \max_{e_j \in \widetilde{\mathcal{E}}} \widetilde{\mathbf{H}}_{ij}}. \tag{9}$$

Since global node-level soft coverage does not explicitly account for regional balance, we further introduce Voronoi-region coverage. For candidate hyperedge e_j, we measure its aggregated incidence strength within each anchor-induced Voronoi region $\mathcal{R}_k$ as $a_{kj} = \sum_{v_i \in \mathcal{R}_k} \widetilde{\mathbf{H}}_{ij}$, and normalize it as $b_{kj} = \frac{a_{kj}}{\max_\ell a_{k\ell}}$. Based on the normalized regional support, the accumulated coverage of Voronoi region $\mathcal{R}_k$ and the overall region-level coverage are defined as

$$y_k(\mathcal{S}) = \min\left\{1, \sum_{e_j \in \mathcal{S}} b_{kj}\right\}, \qquad F_{\text{region}}(\mathcal{S}) = \frac{1}{R_{\text{active}}} \sum_{k=1}^{R_{\text{active}}} y_k(\mathcal{S}), \tag{10}$$

where the active Voronoi regions are re-indexed from 1 to R_{active}. Here, $y_k(\mathcal{S})$ denotes the accumulated coverage of region $\mathcal{R}_k$, capped at one, and R_{active} is the number of Voronoi regions covered by at least one candidate hyperedge.

We combine the node-level and region-level coverage objectives with equal weights. Starting from $\mathcal{S}_0 = \varnothing$, we greedily select at each step the candidate hyperedge that yields the largest marginal increase in the combined coverage score. Let j_m denote the index of the candidate selected at the m-th step, producing the greedy ordering $j_1, j_2, \ldots, j_C$. Accordingly, $\mathcal{S}_m = \{\tilde{e}_{j_1}, \tilde{e}_{j_2}, \ldots, \tilde{e}_{j_m}\}$ denotes the set of the first m selected candidate hyperedges, where $m = 1, \ldots, C$. The combined coverage objective is monotone submodular, and its greedy prefix has a $(1 - 1/e)$ approximation guarantee for any fixed budget (Appendices A.6 and A.7). We then determine the coverage-based hyperedge count from the knee of the combined coverage curve:

$$M_{\text{coverage}} = \max \arg\max_{1 \le m \le C} \left(\frac{1}{2} \left[F_{\text{node}}(\mathcal{S}_m) + F_{\text{region}}(\mathcal{S}_m) \right] - \frac{m}{C} \right). \tag{11}$$

The final number of retained hyperedges is jointly determined by the spectral and coverage criteria:

$$M = \min\{C, \max(1, M_{\text{spectral}}, M_{\text{coverage}})\}, \qquad \mathbf{H} = \widetilde{\mathbf{H}}_{:,\{j_1,\ldots,j_M\}} \in \mathbb{R}^{N \times M}. \tag{12}$$

Since count control selects only cardinality-regulated candidates, the final incidence matrix preserves their capacity bounds (Appendix A.5). Thus, the spectral criterion preserves the dominant structural directions, whereas the coverage criterion favors candidate hyperedges that provide complementary node- and region-level information. The resulting sparse weighted incidence matrix $\mathbf{H}$ is subsequently used for hypergraph message passing.

3.4 Dynamic Hypergraph Message Passing

At dynamic layer ℓ, let $\mathbf{Z}^{(\ell)} \in \mathbb{R}^{N \times d}$ denote the input node representations obtained by the APPNP encoder and $\mathbf{H}^{(\ell)} \in \mathbb{R}^{N \times M^{(\ell)}}$ denote the final incidence matrix, where $\mathbf{Z}^{(0)} = \mathbf{Z}_g$ and $M^{(\ell)}$ is the number of retained hyperedges. We define $\mathbf{D}_v^{(\ell)}$ and $\mathbf{D}_e^{(\ell)}$ as diagonal matrices whose entries are the row sums and column sums of $\mathbf{H}^{(\ell)}$, respectively.

For $M^{(\ell)} > 0$, normalized node-to-hyperedge-to-node propagation and feature transformation are jointly expressed as

$$\mathbf{U}^{(\ell)} = \text{Dropout}\left(\text{GELU}\left(\left(\mathbf{D}_v^{(\ell)}\right)^{-1/2} \mathbf{H}^{(\ell)} \left(\mathbf{D}_e^{(\ell)}\right)^{-1} \left(\mathbf{H}^{(\ell)}\right)^{\top} \left(\mathbf{D}_v^{(\ell)}\right)^{-1/2} \mathbf{Z}^{(\ell)} \mathbf{W}_m^{(\ell)} \right) \mathbf{W}_o^{(\ell)} \right), \tag{13}$$

Table 1: Statistics of the evaluated datasets.

Dataset	Nodes	Edges	Features	Classes	Split
Cora	2,708	5,429	1,433	7	Public
CiteSeer	3,327	4,732	3,703	6	Public
PubMed	19,717	44,338	500	3	Public
Physics	34,493	247,962	8,415	5	50/25/25
Amazon Computers	13,752	245,861	767	10	Fixed
ogbn-arxiv	169,343	1,166,243	128	40	Official
Yelp	716,847	6,977,410	300	100	Official

where GELU refers to the Gaussian Error Linear Unit activation function, $\mathbf{W}_m^{(\ell)}$ and $\mathbf{W}_o^{(\ell)}$ are learnable weight matrices, and $\mathbf{U}^{(\ell)}$ is the resulting hypergraph message.

A learnable gate controls the contribution of this message:

$$g^{(\ell)} = \mathrm{sigmoid}(\alpha^{(\ell)}), \qquad \mathbf{Z}^{(\ell+1)} = \begin{cases} \mathrm{LayerNorm}\left(\mathbf{Z}^{(\ell)}\right), & M^{(\ell)} = 0, \\ \mathrm{LayerNorm}\left(\mathbf{Z}^{(\ell)} + g^{(\ell)}\mathbf{U}^{(\ell)}\right), & M^{(\ell)} > 0. \end{cases} \tag{14}$$

Here, $\alpha^{(\ell)} \in \mathbb{R}$ is a learnable scalar gate parameter for the ℓ-th layer, and $g^{(\ell)}$ is the corresponding residual gate that controls the contribution of the hypergraph-aggregated message $\mathbf{U}^{(\ell)}$. Thus, the model limits the contribution of the reconstructed hypergraph during early training and bypasses hypergraph aggregation when no hyperedge is retained.

Finally, the classifier combines the graph representation with the outputs of all L dynamic layers:

$$\widehat{\mathbf{Y}} = f_{\mathrm{cls}}\left(\mathrm{Dropout}\left[\mathbf{Z}_g \| \mathbf{Z}^{(1)} \| \cdots \| \mathbf{Z}^{(L)}\right]\right), \tag{15}$$

where $\|$ denotes feature concatenation, f_{cls} denotes the learnable classification head, and $\widehat{\mathbf{Y}}$ contains the prediction logits. The computational complexity is analyzed in Appendix B.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate our method on seven node-classification benchmarks: Cora, CiteSeer, PubMed Yang et al. (2016), Physics, Amazon Computers Shchur et al. (2018), ogbn-arxiv Hu et al. (2020), and Yelp Zeng et al. (2020). The first six are single-label datasets, whereas Yelp is a multi-label classification dataset. We use the public splits for Cora, CiteSeer, and PubMed, the official splits for ogbn-arxiv and Yelp, and fixed splits for Physics and Amazon Computers. Detailed dataset statistics and split protocols are provided in Table 1 and Appendix C, respectively. The sensitivity analysis of the key-anchor ratio is provided in Appendix C.

Baselines. We compare with six representative hypergraph neural networks HGNN Feng et al. (2019), HGNN+ Gao et al. (2022), HyperGCN Yadati et al. (2019), HNHN Dong et al. (2020), ED-HNN Wang et al. (2023) and ACF-HNN Huang et al. (2025), as well as four dynamic hypergraph models, including DHGNN Jiang et al. (2019), DHKH Kang et al. (2022), TDHNN Zhou et al. (2023), and DHHNN Mei et al. (2025). All baseline methods are reproduced using their publicly available source code and evaluated using the same dataset splits and evaluation metrics.

4.2 Overall Performance

Table 2 reports the node-classification results on seven benchmarks. AD-DHGNN achieves the highest reported mean on every dataset, demonstrating consistent effectiveness across datasets with different scales and characteristics.

Compared with the conventional ED-HNN, AD-DHGNN performs better on all six datasets for which results are available, with the largest improvement of 11.19 percentage points on ogbn-arxiv.

Table 2: Overall node-classification performance (%). Accuracy is reported on all datasets except Yelp, where Micro-F1 is used. Results are shown as mean ± standard deviation. Green and blue indicate the best and second-best mean results, respectively. "-" denotes an out-of-memory (OOM) run.

Method	Cora	CiteSeer	PubMed	Physics	Amazon Computers	ogbn-arxiv	Yelp
Hypergraph Neural Networks							
HGNN (2019)	79.53±0.32	70.19±0.63	78.17±0.21	96.54±0.08	70.41±1.28	60.91±0.33	–
HyperGCN (2019)	80.62±0.53	68.51±0.99	78.44±0.42	96.36±0.25	67.90±0.72	62.85±0.54	–
HNHN (2020)	72.67±0.65	55.84±2.11	70.37±0.58	96.27±0.31	47.11±0.77	53.27±0.62	–
HGNN+ (2022)	79.49±0.41	70.24±0.42	76.85±0.35	96.17±0.12	71.57±0.94	61.47±0.41	–
ED-HNN (2023)	81.86±0.19	69.42±0.87	77.26±0.44	96.73±0.15	77.72±0.99	59.16±0.48	–
ACF-HNN (2025)	80.42±0.74	68.66±1.74	74.11±0.44	96.44±0.07	71.47±0.51	64.73±0.27	–
Dynamic Hypergraph Neural Networks							
DHGNN (2019)	82.41±0.19	68.61±1.65	74.24±0.68	96.47±0.22	73.61±1.32	–	–
DHKH (2022)	81.72±0.35	67.76±1.28	77.68±0.51	96.14±0.24	75.99±1.86	–	–
TDHNN (2023)	82.77±0.32	65.81±1.23	78.13±0.37	96.52±0.18	69.97±0.93	52.77±0.82	32.20±1.15
DHHNN (2025)	82.84±0.60	70.92±0.26	77.52±0.39	95.67±0.46	67.62±2.88	–	–
AD-DHGNN							
Ours	**83.57±0.13**	**71.22±0.24**	**80.14±0.17**	**97.26±0.11**	**80.72±0.71**	**70.35±0.23**	**50.83±0.17**
Δ	**0.73↑**	**0.30↑**	**1.70↑**	**0.53↑**	**3.00↑**	**5.62↑**	**18.63↑**

This result indicates the benefit of reconstructing task-dependent high-order relations instead of relying only on a predefined hypergraph.

Compared with the dynamic DHHNN, AD-DHGNN achieves consistent gains on all five available comparisons, with the largest improvement of 13.10% on Amazon Computers. DHHNN also encounters OOM on ogbn-arxiv and Yelp, whereas AD-DHGNN completes both evaluations. These results show that dynamic reconstruction alone is insufficient for consistent performance and scalability. By coordinating local cardinality regulation with global count control, AD-DHGNN limits oversized and redundant hyperedges while preserving the flexibility of dynamic reconstruction. Detailed hyperparameter settings are provided in Appendix C.

4.3 Ablation Study

We conduct an ablation study on Cora to validate the adaptive hyperedge cardinality and count controls of AD-DHGNN, and the results are reported in Table 3. *APPNP-only* removes the dynamic hypergraph branch, *Candidate* retains the generated candidates without structural control, *Count-only* and *Cardinality-only* retain only one controller, and *Dual-control* uses the complete model. Additional results on PubMed and ogbn-arxiv are provided in Appendix D.

Count-only substantially reduces the number of retained hyperedges, whereas Cardinality-only limits the size of individual hyperedges. However, neither controller alone regulates both structural dimensions. In contrast, Dual-control jointly controls hyperedge count and cardinality, retaining 22/23 hyperedges with maximum cardinalities of 131/132 while reducing Jaccard and spectral redundancy to 0.110/0.100 and 0.035/0.031, respectively. It also achieves the highest accuracy of $83.57 \pm 0.13\%$. These results support the central design of AD-DHGNN: adaptive cardinality control prevents individual hyperedges from over-expanding, while adaptive count control removes globally redundant candidates. Their combination provides a more compact hypergraph structure together with improved predictive performance.

4.4 Structural and Semantic Information Visualization

To further assess the quality of the hypergraphs constructed by AD-DHGNN, we analyze its structural properties and the semantic information of the learned hypergraph. Figure 2 compares the results of HGNN, TDHNN, and AD-DHGNN on Physics. The top row shows binary node-pair co-membership matrices derived from the final incidence matrices. HGNN produces dense connections within and across classes, while TDHNN forms several nearly saturated blocks containing large groups of nodes. In contrast, AD-DHGNN yields a sparser and more class-aligned structure, with substantially fewer cross-class connections. This pattern is consistent with cardinality control limiting oversized hyperedges and count control removing redundant relations.

Table 3: Ablation results on Cora. Structural statistics are reported as Layer 1/Layer 2. Lower Jaccard and spectral redundancy indicate less overlap. Hyperedge count and cardinality are reported descriptively.

Variant	ACC (%)	Hyperedges	Max cardinality	Jaccard red.	Spectral red.
APPNP-only	79.80±1.31	N/A	N/A	N/A	N/A
Candidate	81.67±1.46	64.0/64.0	1627/1487	0.882/0.878	0.908/0.904
Count-only	82.00±1.23	**6.0/5.7**	1595/1412	0.631/0.641	0.424/0.377
Cardinality-only	82.73±1.02	64.0/64.0	135/137	0.407/0.400	0.287/0.285
Dual-control	**83.57±0.13**	22.0/23.0	**131/132**	**0.110/0.100**	**0.035/0.031**

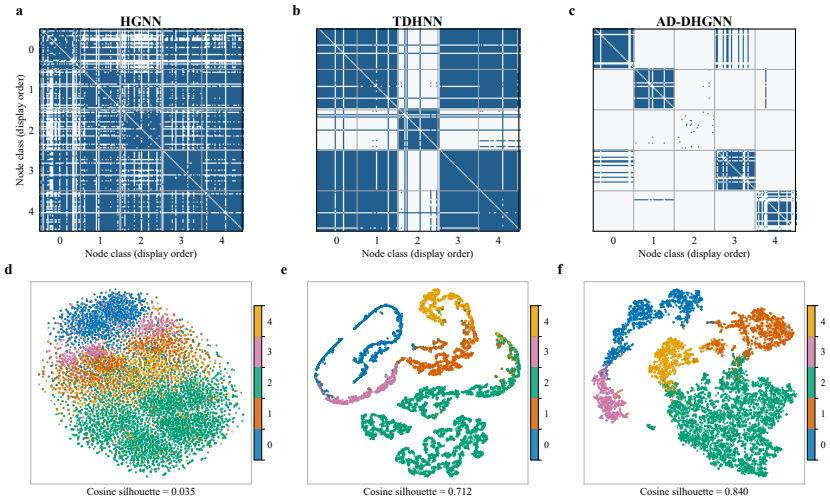

Figure 2: Structural and semantic information visualization over Physics. **a–c**, node-pair co-membership patterns of HGNN, TDHNN, and AD-DHGNN. **d–f**, corresponding node representations with cosine silhouette scores.

The bottom row presents t-distributed stochastic neighbor embedding (t-SNE) projections of the corresponding node representations. HGNN exhibits substantial class overlap, whereas TDHNN provides clearer but still partially mixed class structures. AD-DHGNN produces the most distinct class separation. Its cosine silhouette score reaches 0.840, compared with 0.035 for HGNN and 0.712 for TDHNN, indicating more discriminative node representations.

5 Conclusion

We introduced AD-DHGNN, a dynamic hypergraph neural network that controls both the cardinality and the number of hyperedges. It first uses anchor-induced Voronoi capacities and posterior membership evidence to bound candidate hyperedges, then removes identical supports and applies spectral- and coverage-based criteria to select a compact set for gated message passing. Across seven node-classification benchmarks, AD-DHGNN achieves strong predictive performance while producing compact hypergraphs. Ablations show that cardinality control limits oversized hyperedges, whereas count control reduces redundant candidates. Future work will extend dual-controlled dynamic hypergraph construction to multi-view settings, enabling complementary high-order relations to be adaptively captured and coordinated across views.

GENERATIVE AI USAGE DISCLOSURE

In preparing this work, the authors used OpenAI Codex to assist with Chinese–English translation, manuscript drafting and language editing, organization of related-work material, feedback on the presentation of the methodology and experiments, and the preparation and refinement of plotting code and figure layouts. Generative AI also assisted in organizing and summarizing experimental results, but the authors independently determined the reported results and scientific conclusions.

Generative AI was not used autonomously to formulate the central research idea, determine the final model design, conduct experiments, or alter experimental data. All AI-assisted text, equations, code, figures, references, and interpretations were reviewed and revised by the authors. In particular, the methodological descriptions and equations were checked against the final implementation, and the reported numerical results and visualizations were checked against the original experimental outputs. The authors take full responsibility for the accuracy, originality, and integrity of the manuscript and all associated artifacts.

REFERENCES

Derun Cai, Moxian Song, Chenxi Sun, Baofeng Zhang, Shenda Hong, and Hongyan Li. Hypergraph structure learning for hypergraph neural networks. In *Ijcai*, pp. 1923–1929, 2022.

Eli Chien, Chao Pan, Jianhao Peng, and Olgica Milenkovic. You are allset: A multiset function framework for hypergraph neural networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=hpBTIv2uy_E.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977. doi: 10.1111/j.2517-6161.1977.tb01600.x. URL https://doi.org/10.1111/j.2517-6161.1977.tb01600.x.

Yihe Dong, Will Sawin, and Yoshua Bengio. Hnhn: Hypergraph networks with hyperedge neurons. *arXiv preprint arXiv:2006.12278*, 2020.

Iulia Duta, Giulia Cassarà, Fabrizio Silvestri, and Pietro Lió. Sheaf hypergraph networks. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 12087–12099. Curran Associates, Inc., 2023. doi: 10.52202/075280-0529. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/27f243af2887d7f248f518d9b967a882-Paper-Conference.pdf.

Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3558–3565, 2019. doi: 10.1609/aaai.v33i01.33013558. URL https://doi.org/10.1609/aaai.v33i01.33013558.

Yue Gao, Yifan Feng, Shuyi Ji, and Rongrong Ji. Hgnn+: General hypergraph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3181–3199, 2022.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Changqin Huang, Jialin Sun, Liangliang Zha, Yi Wang, and Xiaodi Huang. Beyond homophily: Adaptive cross-frequency convolution for hypergraph learning. *Knowledge-Based Systems*, pp. 115064, 2025.

Jing Huang and Jie Yang. Unignn: a unified framework for graph and hypergraph neural networks. In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 2563–2569. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/353. URL https://doi.org/10.24963/ijcai.2021/353. Main Track.

Jianwen Jiang, Yuxuan Wei, Yifan Feng, Jingxuan Cao, and Yue Gao. Dynamic hypergraph neural networks. In *IJCAI*, pp. 2635–2641, 2019.

Xiaojun Kang, Xinchuan Li, Hong Yao, Dan Li, Bo Jiang, Xiaoyue Peng, Tiejun Wu, Shihua Qi, and Lijun Dong. Dynamic hypergraph neural networks based on key hyperedges. *Information Sciences*, 616:37–51, 2022.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*, 2017. URL https://mlanthology.org/iclr/2017/kipf2017iclr-semi/.

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=H1gL-2A9Ym.

Zhangyu Mei, Xiao Bi, Dianguo Li, Wen Xia, Fan Yang, and Hao Wu. Dhhnn: a dynamic hypergraph hyperbolic neural network based on variational autoencoder for multimodal data integration and node classification. *Information Fusion*, 119:103016, 2025.

Richard A. Redner and Homer F. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26(2):195–239, 1984. doi: 10.1137/1026034. URL https://doi.org/10.1137/1026034.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *15th European Signal Processing Conference*, pp. 606–610, 2007. URL https://eurasip.org/Proceedings/Eusipco/Eusipco2007/Papers/a5p-h05.pdf.

Ville Satopaa, Jeannie Albrecht, David Irwin, and Barath Raghavan. Finding a Kneedle in a haystack: Detecting knee points in system behavior. In *2011 31st International Conference on Distributed Computing Systems Workshops*, pp. 166–171, 2011. doi: 10.1109/ICDCSW.2011.20. URL https://doi.org/10.1109/ICDCSW.2011.20.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Henry Teicher. Identifiability of finite mixtures. *The Annals of Mathematical Statistics*, 34(4):1265–1269, 1963. doi: 10.1214/aoms/1177703862. URL https://doi.org/10.1214/aoms/1177703862.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *International Conference on Learning Representations*, 2018. URL https://mlanthology.org/iclr/2018/velickovic2018iclr-graph/.

Peihao Wang, Shenghao Yang, Yunyu Liu, Zhangyang Wang, and Pan Li. Equivariant hypergraph diffusion neural operators. In *International Conference on Learning Representations (ICLR)*, 2023.

Wei Xu, Xiaoyi Jiang, Qingzhe Cui, Lixiang Xu, and Xiaofeng Wang. Fission-based dynamic hypergraph neural network. *Pattern Recognition*, 180:113995, 2026. ISSN 0031-3203.

Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. Hypergcn: A new method for training graph convolutional networks on hypergraphs. *Advances in neural information processing systems*, 32, 2019.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pp. 40–48. PMLR, 2016.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. GraphSAINT: Graph Sampling Based Inductive Learning Method. In *International Conference on Learning Representations*, 2020. URL https://mlanthology.org/iclr/2020/zeng2020iclr-graphsaint/.

Zizhao Zhang, Haojie Lin, and Yue Gao. Dynamic hypergraph structure learning. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 3162–3169. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi: 10.24963/ijcai.2018/439. URL https://doi.org/10.24963/ijcai.2018/439.

Peng Zhou, Zongqian Wu, Xiangxiang Zeng, Guoqiu Wen, Junbo Ma, and Xiaofeng Zhu. Totally dynamic hypergraph neural network. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pp. 2476–2483, 2023.

APPENDIX

The appendix provides supplementary analyses and experimental details to support the main text.

- Appendix A provides the theoretical analysis of candidate generation, cardinality regulation, coverage construction, and adaptive count control.
- Appendix B derives the time and memory complexity and reports the measured runtime and GPU memory footprint.
- Appendix C describes the dataset protocols, implementation settings, structural metrics, dynamic hypergraph comparison, and key-anchor-ratio sensitivity.
- Appendix D reports the backbone-only results and additional ablations on PubMed and ogbn-arxiv.
- Appendix E evaluates structural recovery, controller complementarity, robustness, and scalability using synthetic ground truth.

A THEORETICAL ANALYSIS

This section addresses the seven theoretical questions underlying candidate generation, cardinality regulation, and count control. The arguments are stated for an arbitrary but fixed dynamic layer and anchor; layer superscripts are omitted for readability. The fixed anchor set contains K distinct anchors after anchor deduplication, while similarities, posterior probabilities, Voronoi regions, and hyperedges are reconstructed at every dynamic layer.

A.1 WHY THE SIMILARITIES ARE MODELED BY TWO POPULATIONS

For anchor a_k and non-anchor node v_i, introduce a latent binary variable $z_{ik} \in \{0, 1\}$. The event $z_{ik} = 1$ denotes that v_i belongs to the relation-supported population of a_k, whereas $z_{ik} = 0$ denotes the background population. We use the conditional model

$$s_{ik} \mid z_{ik} = 0 \sim \mathcal{N}(\mu_{k,-}, \sigma^2_{k,-}), \qquad s_{ik} \mid z_{ik} = 1 \sim \mathcal{N}(\mu_{k,+}, \sigma^2_{k,+}), \tag{16}$$

with $\Pr(z_{ik} = 1) = \pi_k \in (0, 1)$ and $\mu_{k,+} > \mu_{k,-}$. Marginalizing the latent variable gives

$$f_k(s) = (1 - \pi_k)\phi_{k,-}(s) + \pi_k\phi_{k,+}(s), \tag{17}$$

where $\phi_{k,-}$ and $\phi_{k,+}$ are the two Gaussian densities.

The two populations follow from the binary candidate-membership decision: each node is either supported as a member of the anchor-induced relation or treated as background for that candidate. The interpretation of the higher-mean component as relation-supported is justified by the alignment assumption $\mu_{k,+} > \mu_{k,-}$: supported nodes are expected to be more similar to the anchor. The lower-mean component is consequently treated as background or noise relative to that anchor. This semantic assignment is a modeling assumption, rather than a consequence of fitting an arbitrary two-component mixture. Without mean ordering or external relation labels, the two fitted components are identifiable only up to permutation and cannot be intrinsically named as relation and noise.

In the notation of the main text, $\pi_k = \omega_{k,+}$ is learned by the finite-step EM procedure. It is distinct from the normalized Voronoi-region size $|\mathcal{R}_k|/N$: π_k enters posterior estimation, whereas $|\mathcal{R}_k|$ determines the candidate capacity.

A.2 Why the Posterior Threshold Is $1/2$

The fitted mixture gives the relation posterior

$$p_{ik} = \Pr(z_{ik} = 1 \mid s_{ik}) = \frac{\pi_k \phi_{k,+}(s_{ik})}{\pi_k \phi_{k,+}(s_{ik}) + (1-\pi_k)\phi_{k,-}(s_{ik})} = \sigma(\xi_{ik}), \tag{18}$$

where the posterior log odds are

$$\xi_{ik} = \log \frac{p_{ik}}{1-p_{ik}} = \log \frac{\pi_k \phi_{k,+}(s_{ik})}{(1-\pi_k)\phi_{k,-}(s_{ik})}. \tag{19}$$

Proposition 1 (Bayes decision threshold). Under equal costs for false inclusion and false exclusion, including v_i in the raw candidate is Bayes optimal if and only if $p_{ik} > 1/2$, equivalently $\xi_{ik} > 0$.

Proof. Conditioned on s_{ik}, predicting relation membership has error probability $1 - p_{ik}$, whereas predicting background has error probability p_{ik}. The former decision has smaller conditional risk precisely when $1 - p_{ik} < p_{ik}$, which is equivalent to $p_{ik} > 1/2$. By Eq. (19), this condition is also equivalent to $\xi_{ik} > 0$. At equality, the two decisions have the same conditional risk; the strict rule assigns the ambiguous node to background. Therefore

$$e_{k,\mathrm{raw}} = \{a_k\} \cup \{v_i : p_{ik} > 1/2\} = \{a_k\} \cup \{v_i : \xi_{ik} > 0\}. \tag{20}$$

For unequal false-positive and false-negative costs c_{FP} and c_{FN}, the corresponding Bayes threshold becomes $c_{\mathrm{FP}}/(c_{\mathrm{FP}} + c_{\mathrm{FN}})$. Thus, $1/2$ is the equal-cost decision threshold rather than an additional tuned similarity threshold.

A.3 Upper Bound on the Candidate Capacity

Let $\{\mathcal{R}_k\}_{k=1}^{K}$ be the anchor-induced Voronoi partition. Each of the K distinct anchors is explicitly assigned to its own region, and the candidate-specific capacity is

$$u_k = \max\{2, |\mathcal{R}_k|\}. \tag{21}$$

Proposition 2 (Upper bound on u_k). Assume $N \geq 2$ and $1 \leq K \leq N$. Then

$$2 \leq u_k \leq \max\{2, N - K + 1\} \leq N. \tag{22}$$

Proof. The Voronoi regions partition the N nodes. Because each of the other $K - 1$ regions contains its own anchor, region $\mathcal{R}_k$ can contain at most the remaining $N - (K-1) = N - K + 1$ nodes. Hence $|\mathcal{R}_k| \leq N - K + 1$. Applying the monotone function $r \mapsto \max\{2, r\}$ gives $u_k \leq \max\{2, N - K + 1\}$. The final inequality follows from $N \geq 2$.

A.4 Why the Two-Gaussian Model Can Be Estimated

Proposition 3 (Conditional identifiability and estimability). Suppose the model in Eq. equation 16 is correctly specified for an idealized independent sample sequence, the two component parameter pairs are distinct, the mixture weight is bounded away from zero and one, and the variances lie in $[\sigma^2_{\min}, \sigma^2_{\max}]$ for $0 < \sigma_{\min} < \sigma_{\max} < \infty$. Then the two-Gaussian mixture is identifiable up to component permutation. Ordering the means as $\mu_{k,-} < \mu_{k,+}$ removes this label ambiguity. Under these regularity conditions, a global maximum-likelihood estimator is consistent as the number of observed similarities increases.

Justification. Finite mixtures of univariate Gaussian distributions are identifiable up to label permutation (Teicher, 1963). The parameter bounds exclude vanishing component masses and variance collapse, and mean ordering selects one of the two equivalent label permutations. Standard finite-mixture results then establish consistency of a global maximum-likelihood estimator under correct specification (Redner & Walker, 1984). EM monotonically increases the observed-data likelihood and converges to a stationary point, but finite-step EM need not reach the global maximum (Dempster et al., 1977).

Accordingly, the proposition establishes why the two Gaussian components are statistically estimable under the stated model. It does not assert that every empirical similarity distribution is exactly a two-Gaussian mixture. The implemented finite-step EM estimator, which uses variance flooring and does not impose all of the compact parameter constraints above, is not claimed to inherit the global-MLE consistency result. Similarities derived from a shared graph and jointly trained representations are also not strictly independent; independence is an idealized condition for the classical estimation result.

A.5 Proof of the Cardinality-Regulation Rule

For a valid raw candidate, Eq. (6) retains the source anchor and, when truncation is necessary, selects the $u_k - 1$ non-anchor nodes with the largest posterior log odds.

Proposition 4 (Capacity feasibility and evidence-optimal truncation). Every candidate retained after singleton filtering satisfies

$$2 \le |e_k| \le u_k. \tag{23}$$

If $|e_{k,\mathrm{raw}}| > u_k$, the retained non-anchor set solves

$$e_k \setminus \{a_k\} \in \operatorname*{arg\,max}_{\substack{\mathcal{U} \subseteq e_{k,\mathrm{raw}} \setminus \{a_k\} \\ |\mathcal{U}| = u_k - 1}} \sum_{v_i \in \mathcal{U}} \xi_{ik}. \tag{24}$$

Proof. If $|e_{k,\mathrm{raw}}| \le u_k$, Eq. (6) leaves the candidate unchanged. Singleton filtering supplies the lower bound, so $2 \le |e_k| \le u_k$. If $|e_{k,\mathrm{raw}}| > u_k$, the rule retains one anchor and exactly $u_k - 1$ non-anchor nodes; hence $|e_k| = u_k$.

For optimality, order the non-anchor evidence values as $\xi_{(1)k} \ge \cdots \ge \xi_{(r)k}$. Every feasible set in Eq. (24) contains $u_k - 1$ values, and its sum is at most $\sum_{t=1}^{u_k-1} \xi_{(t)k}$. The TopNodes operation in Eq. (6) attains this upper bound. Ties may produce multiple maximizers but do not change feasibility or the optimal objective value.

A.6 Relationship Between Node and Region Coverage

Let $\widetilde{\mathbf{H}} \in \mathbb{R}_{\ge 0}^{N \times C}$ be the regulated and deduplicated candidate incidence matrix. For $\mathcal{S} \subseteq \{1, \ldots, C\}$, define

$$F_{\mathrm{node}}(\mathcal{S}) = \frac{\sum_{i=1}^{N} \max_{j \in \mathcal{S}} \widetilde{\mathbf{H}}_{ij}}{\sum_{i=1}^{N} \max_{1 \le j \le C} \widetilde{\mathbf{H}}_{ij}}, \tag{25}$$

where the maximum over the empty set is defined as zero. For every active Voronoi region, let $a_{kj} = \sum_{v_i \in \mathcal{R}_k} \widetilde{\mathbf{H}}_{ij}$ and $b_{kj} = a_{kj} / \max_q a_{kq}$. The region-level and combined coverage are

$$\begin{aligned} F_{\mathrm{region}}(\mathcal{S}) &= \frac{1}{R_{\mathrm{active}}} \sum_{k=1}^{R_{\mathrm{active}}} \min\left\{1, \sum_{j \in \mathcal{S}} b_{kj}\right\}, \\ F_{\mathrm{cov}}(\mathcal{S}) &= \frac{F_{\mathrm{node}}(\mathcal{S}) + F_{\mathrm{region}}(\mathcal{S})}{2}. \end{aligned} \tag{26}$$

F_{node} is a micro-averaged measure: each node contributes according to its strongest selected membership, so larger node populations receive more total weight. F_{region} is a macro-averaged measure: each active Voronoi region contributes equally after its attainable support is normalized and capped. They therefore measure complementary aspects of coverage rather than approximating the same quantity.

Proposition 5 (Validity of the combined coverage). For every $\mathcal{S}$,

$$0 \le F_{\mathrm{node}}(\mathcal{S}), F_{\mathrm{region}}(\mathcal{S}), F_{\mathrm{cov}}(\mathcal{S}) \le 1, \tag{27}$$

and all three functions equal zero at $\varnothing$ and one at the complete candidate set. Moreover,

$$\min\{F_{\mathrm{node}}(\mathcal{S}), F_{\mathrm{region}}(\mathcal{S})\} \le F_{\mathrm{cov}}(\mathcal{S}) \le \max\{F_{\mathrm{node}}(\mathcal{S}), F_{\mathrm{region}}(\mathcal{S})\}, \tag{28}$$

and

$$F_{\mathrm{cov}}(\mathcal{S}) \ge 1 - \varepsilon \quad \Longrightarrow \quad \min\{F_{\mathrm{node}}(\mathcal{S}), F_{\mathrm{region}}(\mathcal{S})\} \ge 1 - 2\varepsilon. \tag{29}$$

Proof. The numerator of F_{node} is bounded by its complete-set denominator. Each capped regional contribution belongs to $[0, 1]$. At the complete candidate set, every node attains its maximum incidence and every active region contains a candidate attaining its normalizing maximum; hence both coverage terms equal one. Equations (28) and (29) follow from taking the arithmetic mean of two values in $[0, 1]$.

Both component functions are also monotone and submodular. For a node i, the marginal gain of adding candidate e is

$$\max\{0, \widetilde{\mathbf{H}}_{ie} - \max_{j \in \mathcal{S}} \widetilde{\mathbf{H}}_{ij}\}, \tag{30}$$

which cannot increase as $\mathcal{S}$ grows. For a region k, $\sum_{j \in \mathcal{S}} b_{kj}$ is a nonnegative modular function, and the concave nondecreasing cap $\min\{1, x\}$ gives diminishing returns. Positive normalization and averaging preserve monotonicity and submodularity. Consequently, F_{cov} is a normalized coverage objective that balances global node support with region-level balance. The equal weight $1/2$ is a design choice assigning the same importance to the two normalized objectives.

A.7 Existence of the Spectral and Coverage Knees

Normalize every nonzero column of $\widetilde{\mathbf{H}}$ to unit ℓ_2 norm and let $\mathbf{G} = \overline{\mathbf{H}}^{\top} \overline{\mathbf{H}}$. For any vector $\mathbf{x}$,

$$\mathbf{x}^{\top} \mathbf{G} \mathbf{x} = \|\overline{\mathbf{H}} \mathbf{x}\|_2^2 \geq 0, \tag{31}$$

so $\mathbf{G}$ is positive semidefinite. Its diagonal entries equal one; therefore, for eigenvalues $\lambda_1 \geq \cdots \geq \lambda_C \geq 0$,

$$\sum_{r=1}^{C} \lambda_r = \mathrm{tr}(\mathbf{G}) = C. \tag{32}$$

Define

$$S_m = \frac{1}{C} \sum_{r=1}^{m} \lambda_r, \qquad D_{\mathrm{spec}}(m) = S_m - \frac{m}{C}, \qquad S_0 = D_{\mathrm{spec}}(0) = 0. \tag{33}$$

Its first difference is

$$D_{\mathrm{spec}}(m) - D_{\mathrm{spec}}(m-1) = \frac{\lambda_m - 1}{C}. \tag{34}$$

Because the eigenvalues are nonincreasing, these differences are nonincreasing, so D_{spec} is discrete concave. A maximum exists on the finite domain $\{1, \ldots, C\}$, and its latest maximizer is

$$M_{\mathrm{spectral}} = \max\{m : \lambda_m \geq 1\}. \tag{35}$$

If the spectrum is nonuniform, its mean-one property implies that some eigenvalues lie above one and some below one, so the latest maximizer is an interior index. If every eigenvalue equals one, the curve is linear and the conservative latest-maximizer rule returns C.

For the coverage knee, let $j_1, \ldots, j_C$ be the complete greedy ordering for F_{cov}, let $\mathcal{S}_m = \{j_1, \ldots, j_m\}$, and define

$$\delta_m = F_{\mathrm{cov}}(\mathcal{S}_m) - F_{\mathrm{cov}}(\mathcal{S}_{m-1}). \tag{36}$$

At step m, the greedy rule chooses the largest available marginal gain. Submodularity ensures that every remaining marginal gain can only decrease after a candidate is added. Therefore $\delta_1 \geq \cdots \geq \delta_C \geq 0$. Since $F_{\mathrm{cov}}(\varnothing) = 0$ and the complete candidate set has coverage one, $\sum_{m=1}^{C} \delta_m = 1$. Define

$$D_{\mathrm{cov}}(m) = F_{\mathrm{cov}}(\mathcal{S}_m) - \frac{m}{C}, \qquad D_{\mathrm{cov}}(0) = 0. \tag{37}$$

Then

$$D_{\mathrm{cov}}(m) - D_{\mathrm{cov}}(m-1) = \delta_m - \frac{1}{C}. \tag{38}$$

These differences are nonincreasing, so D_{cov} is also discrete concave and has a latest maximizer

$$M_{\mathrm{coverage}} = \max\{m : \delta_m \geq 1/C\}. \tag{39}$$

If the gains are nonuniform, the latest maximizer is an interior index. If all gains equal $1/C$, the curve is linear and the conservative rule returns C. Thus, both knee maxima always exist; a strict

Table 4: Runtime and memory footprint of dynamic hypergraph methods on Cora. Times are in seconds. Peak training GPU memory is in MiB. Total training time depends on the number of completed epochs.

Method	Total training time (s)	Time per epoch (s)	Inference time (s)	Trainable parameters	Peak GPU memory (MiB)
DHGNN	342.911	13.697	6.347	16,175,948	490.439
DHKH	687.951	13.747	8.545	16,175,948	487.939
TDHNN	**5.559**	**0.015**	**0.005**	**250,503**	**48.033**
DHHNN	3627.621	13.048	6.524	533,008	167.860
AD-DHGNN	11.317	0.172	0.082	326,610	68.526

interior knee exists precisely when the corresponding spectral contributions or greedy gains are nonuniform.

The preceding characterizations use exact arithmetic. The implementation uses floating-point clipping and a late-tie tolerance of $32\epsilon_{\mathrm{mach}}$. Up to this numerical tolerance, it returns the latest maximizer of each knee curve.

B Computational Complexity

Let N be the number of nodes, K the effective anchor count, d the hidden dimension, T_{EM} the number of EM iterations, and C the number of regulated and deduplicated candidates in one dynamic layer. We denote the number of dynamic layers by L, the number of APPNP propagation steps by T_{APPNP}, and the input-graph edge set by $\mathcal{E}_g$. The number of nonzero entries in the retained incidence matrix is $\mathrm{nnz}(\mathbf{H})$.

Time complexity. Node–anchor inference, mixture estimation, and cardinality regulation require $O(NKd + T_{\mathrm{EM}}NK + KN\log N)$ time. Constructing the candidate Gram matrix, computing its spectrum, and performing direct coverage-aware greedy selection require $O(NC^2 + C^3 + C^2(N + K))$ time. Sparse hypergraph propagation and the learned feature transformations add $O(\mathrm{nnz}(\mathbf{H})d + Nd^2)$ time. Therefore, one dynamic layer has time complexity

$$O\big(NKd + T_{\mathrm{EM}}NK + KN\log N + NC^2 + C^3 + C^2(N + K) + \mathrm{nnz}(\mathbf{H})d + Nd^2\big). \quad (40)$$

For L dynamic layers, the corresponding layer-specific costs are summed. The sparse APPNP propagation additionally requires $O(T_{\mathrm{APPNP}}|\mathcal{E}_g|d)$ time.

Space complexity. The structural working-memory requirement of one dynamic layer is

$$O(NK + NC + C^2), \quad (41)$$

which accounts for node–anchor quantities, the candidate incidence matrix, and the candidate Gram matrix. This bound excludes encoder activations, model parameters, gradients, and optimizer states.

B.1 Runtime and Memory Footprint

Table 4 compares the runtime, trainable parameter count, and peak training GPU memory of five dynamic hypergraph methods on Cora. Because total training time depends on the number of completed epochs, time per epoch and inference time provide more direct measures of execution cost.

TDHNN is the fastest method, requiring 0.015 s per epoch and 0.005 s for inference. Its small hyperedge budget and lightweight hypergraph convolution help explain this low cost Zhou et al. (2023). The structural comparison in Figure 3 shows that its few effective hyperedges span large node sets. AD-DHGNNrequires 0.172 s per epoch and 0.082 s for inference. Its additional node–anchor membership estimation, cardinality regulation, and count selection are consistent with the higher cost relative to TDHNN. It still uses less time per epoch and less peak GPU memory than DHGNN, DHKH, and DHHNN.

Table 5: Dataset-specific hyperparameters selected on the validation set for the seven datasets. The anchor percentage is $100K_0/N$. "Epochs/patience" gives the maximum number of training epochs and the early-stopping patience.

Dataset	K_0 (%)	$K_{\mathrm{APPNP}}/\alpha_{\mathrm{APPNP}}$	Hidden	Dropout	η	ρ_{back}	Weight decay	Epochs/patience
Cora	64 (2.363)	5/0.1	64	0.5	0.01	0.2	0.0007	300/50
CiteSeer	58 (1.743)	5/0.1	128	0.5	0.01	1.0	0.0005	300/50
PubMed	395 (2.003)	10/0.1	64	0.3	0.01	0.2	0.0005	100/30
Physics	173 (0.502)	5/0.3	64	0.5	0.01	0.2	0.0005	100/30
Amazon Computers	241 (1.752)	5/0.1	64	0.5	0.01	0.5	0.0005	300/50
ogbn-arxiv	424 (0.250)	10/0.1	128	0.3	0.01	0.2	0.0005	600/100
Yelp	1218 (0.170)	10/0.1	64	0.3	0.01	0.2	0.0005	600/50

Table 6: Validation-selected checkpoints from the dataset-specific hyperparameter search. Accuracies are reported for the corresponding single selection run.

Dataset	Best epoch	Val. Acc. (%)	Test Acc. (%)
Cora	18	81.20	83.60
CiteSeer	7	69.80	71.70
PubMed	28	79.40	80.10
Physics	97	97.1240	97.1011
Amazon Computers	49	91.3333	80.6973
ogbn-arxiv	590	71.9722	70.3537

C Detailed Experimental Settings

Dataset protocols. Cora, CiteSeer, and PubMed use the public Planetoid splits. Physics uses a fixed stratified $50\%/25\%/25\%$ training, validation, and test split with split seed 0. Amazon Computers uses 20 training nodes and 30 validation nodes per class, with the remaining nodes used for testing. ogbn-arxiv uses the official Open Graph Benchmark chronological split and its original signed continuous features. Yelp uses its official masks and is treated as a multi-label node-classification dataset. All experiments use the transductive setting.

Anchor budget and construction configuration. The initial anchor budget K_0 is a supplied hyperparameter rather than a function of the number of nodes. Table 5 reports the selected configurations currently available for the six single-label datasets. The main AD-DHGNN configuration follows the same construction order as the main text: candidate generation, candidate-specific cardinality regulation, exact support deduplication, and adaptive count control. The count controller combines a spectral knee with a coverage knee and uses a coverage-aware greedy ordering to determine which candidate hyperedges are retained. We denote the number of APPNP propagation steps by K_{APPNP}, the APPNP teleport probability by α_{APPNP}, the base learning rate by η, and the learning-rate multiplier applied to the APPNP backbone by ρ_{back}. Across datasets, we use two dynamic hypergraph layers, six EM iterations, a variance floor of 10^{-4}, a posterior membership threshold of 0.5, a minimum hyperedge length of two, and an initial message-gate logit of -2.

For traceability, Table 6 records the available validation-selected checkpoints from the hyperparameter search. These single-run values document model selection and are not used in place of the multi-seed test results reported in the main comparison.

Software and hardware. All experiments were run with Python 3.9.18 and PyTorch 2.5.1+cu121 under CUDA 12.1 on an NVIDIA A800 GPU with 80 GB of memory.

Evaluation details. For the six single-label datasets, we report node-classification accuracy. For Yelp, we optimize binary cross-entropy with logits and report Micro-F1 at a decision threshold of 0.5. Unless otherwise stated, final results are reported as mean $\pm$ sample standard deviation over five model seeds. For each seed, the test set is evaluated once after restoring the checkpoint with the best validation result.

When one table cell contains two values separated by a slash, the left and right values correspond to dynamic layers 1 and 2, respectively. Quantities aggregated across layers are explicitly labeled as

sums and are not compared directly with per-layer values. Runtime reporting includes hypergraph construction time, time per epoch, total training time, and peak GPU memory.

C.1 Structural Evaluation Metrics

We use Jaccard redundancy and spectral redundancy to evaluate overlap among the retained hyperedges. Both metrics are computed from the final weighted incidence matrix $\mathbf{H}^{(\ell)} \in \mathbb{R}^{N \times M_\ell}$ at dynamic layer ℓ and are used only for evaluation. Let

$$\mathbf{B}^{(\ell)} = \mathbb{I}[\mathbf{H}^{(\ell)} > 0], \qquad \Omega_m^{(\ell)} = \{v_i : B_{im}^{(\ell)} = 1\}, \tag{42}$$

where $\Omega_m^{(\ell)}$ is the binary node support of hyperedge m.

Jaccard redundancy. For two distinct hyperedges, their support overlap is

$$J_{mn}^{(\ell)} = \frac{|\Omega_m^{(\ell)} \cap \Omega_n^{(\ell)}|}{|\Omega_m^{(\ell)} \cup \Omega_n^{(\ell)}|}. \tag{43}$$

For each hyperedge, we retain its largest overlap with any competing hyperedge and then average these values:

$$R_{\text{Jac}}^{(\ell)} = \frac{1}{M_\ell} \sum_{m=1}^{M_\ell} \max_{n \neq m} J_{mn}^{(\ell)}. \tag{44}$$

We set $R_{\text{Jac}}^{(\ell)} = 0$ when $M_\ell = 1$ and report it as not applicable when $M_\ell = 0$. This metric lies in $[0, 1]$. A larger value means that a typical retained hyperedge has a near-duplicate support, whereas a smaller value indicates lower local overlap. It does not measure whether the memberships are semantically correct.

Spectral redundancy. Jaccard redundancy measures pairwise overlap using binary supports. To characterize global linear dependence among the weighted incidence patterns, we first remove any zero columns from the final incidence matrix. Let M_ℓ denote the resulting number of non-empty hyperedges, and let $\overline{\mathbf{H}}^{(\ell)} \in \mathbb{R}^{N \times M_\ell}$ contain their unit-ℓ_2 normalized weighted incidence columns. Empty hyperedges, when present in a baseline construction, are counted separately and are not included in the following spectral calculation. We define the final Gram matrix as

$$\mathbf{G}_{\text{final}}^{(\ell)} = (\overline{\mathbf{H}}^{(\ell)})^\top \overline{\mathbf{H}}^{(\ell)}. \tag{45}$$

Its entry measures the cosine similarity between two weighted incidence columns. Let $\eta_1^{(\ell)}, \ldots, \eta_{M_\ell}^{(\ell)}$ be the eigenvalues of $\mathbf{G}_{\text{final}}^{(\ell)}$. Following the entropy-based effective rank Roy & Vetterli (2007), their normalized spectral masses and effective rank are

$$\psi_r^{(\ell)} = \frac{\eta_r^{(\ell)}}{\sum_{q=1}^{M_\ell} \eta_q^{(\ell)}}, \qquad r_{\text{eff}}(\mathbf{H}^{(\ell)}) = \exp\left(- \sum_{r : \psi_r^{(\ell)} > 0} \psi_r^{(\ell)} \log \psi_r^{(\ell)} \right). \tag{46}$$

The spectral redundancy of the final non-empty hyperedge set is

$$R_{\text{spec}}^{(\ell)} = 1 - \frac{r_{\text{eff}}(\mathbf{H}^{(\ell)})}{M_\ell}. \tag{47}$$

Because

$$1 \leq r_{\text{eff}}(\mathbf{H}^{(\ell)}) \leq \text{rank}(\mathbf{H}^{(\ell)}) \leq \min(N, M_\ell), \tag{48}$$

the metric satisfies

$$\max\left(0, 1 - \frac{N}{M_\ell}\right) \leq R_{\text{spec}}^{(\ell)} \leq 1 - \frac{1}{M_\ell}. \tag{49}$$

When $M_\ell \leq N$, the lower bound of zero is attained when the normalized incidence columns are mutually orthogonal. When $M_\ell > N$, zero spectral redundancy is unattainable because the incidence matrix can contain at most N independent column directions. The upper bound is attained when all normalized columns are identical. If $M_\ell = 0$, the metric is undefined and is reported as not applicable. Effective rank is used here only as an evaluation diagnostic and is distinct from the candidate-set spectral quantity used by the count controller. For both Jaccard and spectral redundancy, lower values indicate less redundant final hypergraphs.

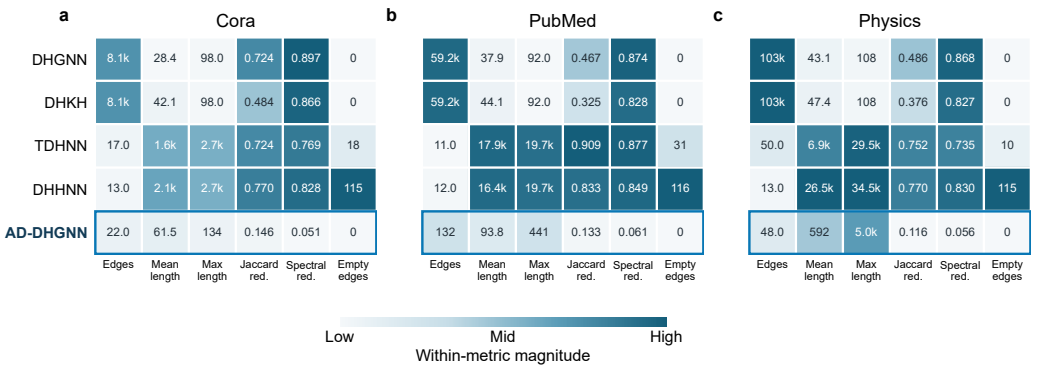

Figure 3: Structural profiles of dynamic hypergraph constructions on **a** Cora, **b** PubMed, and **c** Physics. Rows denote methods and columns report effective hyperedge count, mean and maximum cardinality, Jaccard redundancy, spectral redundancy, and empty-hyperedge count. Cell labels show the measured values, with "k" denoting thousands. Color intensity is normalized independently for each metric across all panels, with logarithmic scaling for count and cardinality metrics. Darker cells indicate larger values.

C.2 Structural Comparison with Dynamic Hypergraph Methods

We compare the structures produced by four representative dynamic hypergraph methods and AD-DHGNN on Cora, PubMed, and Physics. Figure 3 reports effective hyperedge count, mean and maximum cardinality, Jaccard and spectral redundancy, and empty-hyperedge count. For AD-DHGNN, the second dynamic layer is reported.

The compared methods exhibit clearly different structural patterns. DHGNN and DHKH construct many short hyperedges, whereas TDHNN and DHHNN retain far fewer but much larger hyperedges and also produce empty ones. AD-DHGNN instead keeps 22, 132, and 48 effective hyperedges on Cora, PubMed, and Physics, respectively, with no empty hyperedges and consistently low Jaccard and spectral redundancy. These results show that the proposed dual control avoids both excessive hyperedge proliferation and over-expanded hyperedges, producing a more compact and balanced dynamic structure.

C.3 Sensitivity to the Key-Anchor Ratio

We study the sensitivity of AD-DHGNN to the key-anchor ratio, which determines the number of anchors used for dynamic hypergraph construction. As shown in Figure 4, performance generally peaks at an intermediate ratio. The best settings are 1.00%, 2.00%, 1.75%, 0.50%, 1.75%, and 0.25% on Cora, PubMed, CiteSeer, Physics, Amazon Computers, and ogbn-arxiv, respectively. This pattern is consistent with the role of anchors, since too few anchors may limit the diversity of candidate relations, whereas too many may introduce redundant candidates and additional structural cost. The relatively small variation on Cora further indicates that the method is stable within a reasonable neighborhood of its selected ratio. Overall, a key-anchor ratio of 0.25%–2.00% provides a practical default range across the evaluated datasets. The Yelp sweep is omitted because the evaluated configurations exceeded the available GPU memory and resulted in out-of-memory (OOM) errors.

D Additional Ablation Results

The main text reports the Cora ablation. Here, we report the APPNP-only reference across seven datasets and extend the structural ablation to PubMed and ogbn-arxiv. The two larger graphs test whether count and cardinality control address distinct structural failures beyond Cora.

Variants and reporting protocol. We compare five configurations. *APPNP-only* removes the dynamic hypergraph branch and retains only the Approximate Personalized Propagation of Neural

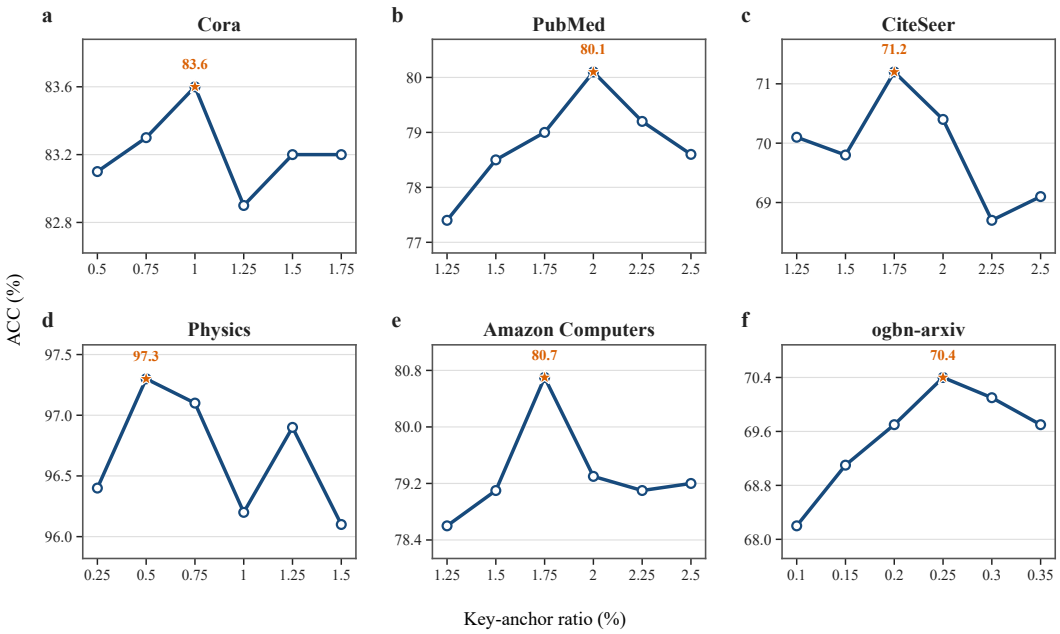

Figure 4: Sensitivity to the key-anchor ratio on six datasets. The horizontal axis denotes the ratio of key anchors and the vertical axis reports classification accuracy (ACC). Stars mark the best setting for each dataset. Dataset-specific vertical ranges are used to make within-dataset variations visible.

Predictions (APPNP) encoder. *Candidate* constructs and deduplicates candidate hyperedges but applies neither structural controller. *Count-only* skips candidate cardinality regulation and applies the spectral-and-coverage count controller to the deduplicated raw candidates. *Cardinality-only* applies the anchor-induced Voronoi capacity rule before exact support deduplication and retains all resulting candidates. *Dual-control* applies candidate cardinality regulation, exact support deduplication, and then spectral-and-coverage count control, matching the complete AD-DHGNN pipeline.

Accuracy is reported as mean $\pm$ sample standard deviation across model seeds. Each structural entry reports the seed-averaged value at dynamic layers 1 and 2, separated by a slash. Hyperedge count describes the number of final incidence columns. Maximum cardinality is the size of the largest retained hyperedge. Jaccard and spectral redundancy quantify local overlap and global linear dependence, respectively. Count and maximum cardinality should be interpreted together. Lower redundancy indicates less repeated structure. "N/A" indicates that a quantity is not applicable because no hypergraph is constructed, and "–" indicates an unavailable result.

D.1 APPNP-Only Baseline Across Datasets

Table 7 reports the performance obtained when the dynamic hypergraph branch is removed. This experiment isolates the APPNP encoder and provides the backbone-only reference for the subsequent ablations.

The APPNP-only results provide a reference for the contribution of dynamic hypergraph message passing. Its performance varies across datasets, with a particularly large standard deviation on Amazon Computers ($34.59 \pm 24.18\%$). This configuration constructs no hypergraph, so the structural comparisons below concern the other variants.

D.2 PubMed

On PubMed (Table 8), Candidate retains about 296 hyperedges per layer. Its largest hyperedges contain more than 12,000 nodes, and both redundancy measures approach one. Count-only reduces the count to 6.6 but leaves maximum cardinality above 11,800 nodes. Cardinality-only limits the maximum to 602.8/571.6 nodes but retains about 296 hyperedges. Dual-control retains 92.0/89.8

Table 7: Performance of the APPNP-only baseline across seeds 0–4. Accuracy is reported on all datasets except Yelp, where Micro-F1 is used. Values are mean ± sample standard deviation in percent.

Dataset	Test performance (%)
Cora	79.80 ± 1.31
CiteSeer	65.56 ± 0.34
PubMed	78.30 ± 0.17
Physics	92.37 ± 3.17
Amazon Computers	34.59 ± 24.18
ogbn-arxiv	67.44 ± 0.17
Yelp	21.52 ± 1.88

Table 8: Ablation results on PubMed. Structural statistics are reported as dynamic layer 1/dynamic layer 2.

Variant	Accuracy (%)	Hyperedges	Max cardinality	Jaccard red.	Spectral red.
APPNP-only	78.30 ± 0.17	N/A	N/A	N/A	N/A
Candidate	78.42 ± 0.24	295.8/295.8	12364.2/12421.8	0.985/0.985	0.989/0.989
Count-only	78.71 ± 0.31	$\mathbf{6.6/6.6}$	12172.2/11861.6	0.748/0.766	0.544/0.542
Cardinality-only	79.14 ± 0.33	295.6/295.8	$\mathbf{602.8/571.6}$	0.332/0.343	0.326/0.341
Dual-control	$\mathbf{80.14 \pm 0.17}$	92.0/89.8	620.0/581.6	$\mathbf{0.137/0.144}$	$\mathbf{0.056/0.062}$

hyperedges with maximum cardinality 620.0/581.6 and reduces Jaccard and spectral redundancy to 0.137/0.144 and 0.056/0.062. It reaches $80.14 \pm 0.17\%$ accuracy, exceeding APPNP-only, Candidate, Count-only, and Cardinality-only by 1.84, 1.72, 1.43, and 1.00 percentage points, respectively.

D.3 OGBN-ARXIV

On ogbn-arxiv (Table 9), Count-only retains 11.6/11.0 hyperedges, but the maximum cardinality exceeds 83,000 nodes in both layers. Cardinality-only reduces the maxima to 9,541.2/8,589.0 nodes while retaining 423.2 hyperedges per layer. Dual-control retains 118.6/117.8 hyperedges with maxima of 9,556.0/8,570.0 nodes and achieves the lowest available Jaccard and spectral redundancy, 0.158/0.157 and 0.064/0.064. Its $70.35 \pm 0.23\%$ accuracy exceeds APPNP-only, Count-only, and Cardinality-only by 2.91, 1.02, and 1.07 percentage points, respectively. The Candidate result is unavailable on this dataset.

D.4 CROSS-DATASET FINDINGS

Across both datasets, Count-only removes most hyperedges but leaves very large supports. Cardinality-only bounds their size but retains nearly all candidates. Dual-control balances these dimensions and attains the highest mean accuracy and lowest reported redundancy. Its maximum cardinality is close to, but not always lower than, that of Cardinality-only. Because the variants are trained separately, their maxima compare different learned candidate sets; the table does not test whether one variant's final hyperedges form a subset of another's. The reported means and standard deviations support a descriptive comparison, while statistical significance would require paired tests on the underlying seed-level results.

E SYNTHETIC EXPERIMENTS WITH STRUCTURAL GROUND TRUTH

Real graphs do not reveal their ground-truth hyperedges. We therefore use controlled synthetic data to test whether AD-DHGNN recovers the latent supports, removes redundant candidates, and enforces adaptive cardinality bounds. These experiments evaluate the construction mechanism directly and complement, rather than replace, the node-classification results in the main text.

Table 9: Ablation results on ogbn-arxiv. Structural statistics are reported as dynamic layer 1/dynamic layer 2.

Variant	Accuracy (%)	Hyperedges	Max cardinality	Jaccard red.	Spectral red.
APPNP-only	67.44 ± 0.17	N/A	N/A	N/A	N/A
Candidate	–	–	–	–	–
Count-only	69.33 ± 0.29	$\mathbf{11.6/11.0}$	83977.4/86763.2	0.509/0.506	0.440/0.426
Cardinality-only	69.28 ± 0.20	423.2/423.2	$\mathbf{9541.2}$/8589.0	0.368/0.367	0.356/0.355
Dual-control	$\mathbf{70.35 \pm 0.23}$	118.6/117.8	9556.0/$\mathbf{8570.0}$	$\mathbf{0.158/0.157}$	$\mathbf{0.064/0.064}$

Table 10: Synthetic ablation in the reference setting. Higher is better for support F1 and coverage; lower is better for length MAPE and redundancy. The "above capacity" column is a diagnostic for variants without length control and a constraint check for variants with length control.

Variant	Final M	Support F1	Length MAPE	Coverage	R_{Jac}	R_{spec}	Above capacity
Candidate	20.60 ± 0.55	0.9931 ± 0.0030	0.0119	0.9966	0.0616	0.037668	20.6
Count-only	20.00 ± 0.00	0.9927 ± 0.0032	0.0128	0.9966	0.0067	0.000123	20.0
Cardinality-only	20.60 ± 0.55	0.9713 ± 0.0094	0.0507	0.9286	0.0194	0.008199	0
Dual-control	20.00 ± 0.00	0.9726 ± 0.0084	0.0483	0.9310	0.0046	0.000079	0

E.1 Experimental protocol

The reference setting contains $N = 1{,}000$ nodes and $M^\star = 20$ latent hyperedges with a mean length of 50. Relation and background similarities are sampled from two Gaussian components with standardized separation $D = 6$. Unless stated otherwise, each result is averaged over five random seeds. We compare four variants constructed from the same posterior candidates: *Candidate* retains all valid candidates after exact support deduplication. *Count-only* skips cardinality regulation and applies the spectral-and-coverage count controller to the deduplicated raw candidates. *Cardinality-only* applies the adaptive capacity constraint before exact support deduplication and retains all resulting candidates. *Dual-control* applies adaptive cardinality regulation, exact support deduplication, and then the spectral-and-coverage count controller. The matched support F1 pairs predicted and true hyperedges by maximum-weight bipartite matching. We also report the final hyperedge count, length mean absolute percentage error (MAPE), node coverage, the number of nonzero incidences, Jaccard redundancy, spectral redundancy, and the number of hyperedges above their inferred capacity. Values are shown as mean $\pm$ standard deviation only when the supplied aggregate contains both quantities.

E.2 Recovery and controller complementarity

Table 10 isolates the roles of the two controllers. Candidate generation alone recovers the latent supports almost perfectly, but leaves 20.6 hyperedges above their inferred capacities and retains substantially more redundancy. Count control reduces Jaccard redundancy from 0.0616 to 0.0067 and spectral redundancy from 0.037668 to 0.000123 while preserving support F1. Length control eliminates all capacity exceedances, with a moderate reduction in support F1 and coverage. Combining both controllers retains exactly 20 hyperedges, removes all capacity exceedances, and gives the lowest redundancy. Its support F1 of 0.9726 shows that this structural regularization is not lossless, but the retained supports remain close to the ground truth.

E.3 Count control under structural complexity

We first vary the true number of hyperedges. Dual-control tracks $M^\star \in \{10, 20, 40, 80\}$, returning 10.0, 20.0, 40.0, and 78.8 hyperedges, respectively (Table 11). Support F1 declines gradually from 0.9853 to 0.9266 as the number of latent structures increases, while the capacity constraint remains satisfied throughout.

We next increase the number R of proposals generated for each of 20 latent hyperedges (Table 12). Exact deduplication alone cannot remove near duplicates: at $R = 10$, 200 raw candidates collapse

Table 11: Count recovery by Dual-control as the true number of hyperedges $M^\star$ increases. Count error is the absolute error $|M - M^\star|$.

$M^\star$	Final M	Count error	Support F1	Above capacity
10	10.0	0.0	0.9853	0
20	20.0	0.0	0.9785	0
40	40.0	0.0	0.9622	0
80	78.8	1.2	0.9266	0

Table 12: Effect of near-duplicate proposals. Effective rank is reported as a diagnostic of the deduplicated incidence spectrum; the final count is produced by the spectral-and-coverage controller. Count error is $|M - M^\star|$ with $M^\star = 20$.

R	Raw	Variant	Deduplicated	r_{eff}	Final M	Count error	Support F1	$R_{\text{Jac}}/R_{\text{spec}}$
1	20	Candidate	20.0	n/a	20.0	0.0	0.9913	0.0085/0.0002
1	20	Count-only	20.0	19.996	20.0	0.0	0.9913	0.0085/0.0002
3	60	Candidate	37.2	n/a	37.2	17.2	0.9951	0.7839/0.4916
3	60	Count-only	37.2	18.874	19.2	0.8	0.9498	0.0088/0.0002
5	100	Candidate	49.8	n/a	49.8	29.8	0.9959	0.8837/0.6277
5	100	Count-only	49.8	18.465	18.8	1.2	0.9275	0.0096/0.0002
10	200	Candidate	74.4	n/a	74.4	54.4	0.9972	0.9503/0.7556
10	200	Count-only	74.4	18.178	18.6	1.4	0.9163	0.0105/0.0003

to 74.4 distinct supports, and Candidate retains all 74.4. Their Jaccard and spectral redundancies rise to 0.9503 and 0.7556. In contrast, Count-only retains 18.6 hyperedges, giving an absolute count error of 1.4 while keeping both redundancy measures near zero. Its support F1 decreases to 0.9163. Candidate has higher matched F1 because the matching score does not penalize its 54.4 surplus hyperedges; count and redundancy must therefore be reported together with support recovery.

Figure 5 summarizes count recovery, redundancy removal, and the low-contamination regime.

E.4 Adaptive length allocation

To separate adaptive allocation from merely imposing a global cardinality budget, we compare Dual-control with a variant that applies one uniform cap to every candidate before the same count controller. Both variants have almost identical redundancy, so their difference is not caused by count selection. Adaptive capacities improve support F1 from 0.9359 to 0.9726, halve length MAPE from 0.1061 to 0.0483, and increase coverage from 0.8416 to 0.9310 (Table 13). This result supports anchor-specific allocation rather than a shared cap.

Across target mean lengths from 10 to 100, Count-only achieves lower length error but exceeds the inferred capacity for all 20 hyperedges. Dual-control eliminates these exceedances and reduces incidence size by 5.5% to 7.7%, at the cost of lower coverage and slightly higher length error (Table 14). The Cardinality controller should therefore be interpreted as a conservative structural constraint, not as an unconditionally lossless length estimator.

Figure 6 summarizes the adaptive length allocation results and the corresponding construction-time and memory profiles.

E.5 Separation, contamination, and failure boundary

Posterior candidate recovery depends on separable similarity evidence. When D increases from 2 to 8, Count-only support F1 rises from 0.3568 to 0.9999 and its length MAPE falls from 0.6867 to 0.0002 (Table 15). Dual-control follows the same trend while maintaining equal or lower Jaccard redundancy. At $D = 2$, however, both variants fail to recover reliable supports. This establishes a

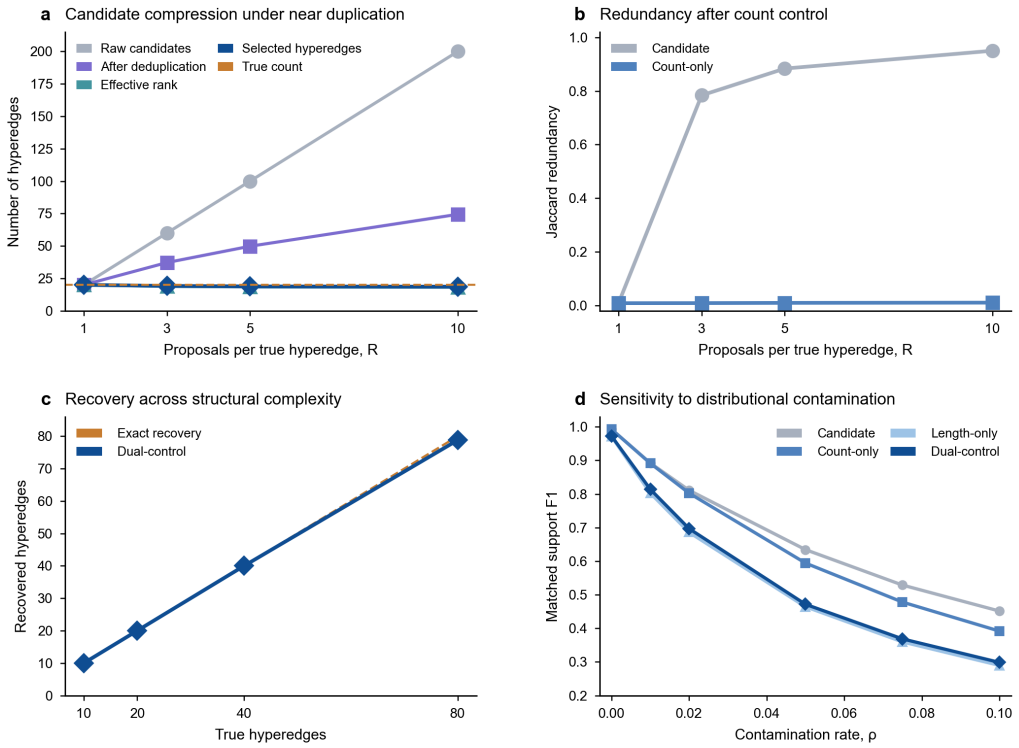

Figure 5: Count control and robustness on synthetic data. **a**, Raw, deduplicated, and retained candidate counts as the number of proposals per latent hyperedge R increases. **b**, Jaccard redundancy with and without count control. **c**, recovered versus true hyperedge count for Dual-control. The dashed line denotes exact recovery. **d**, support F1 under low contamination for the four construction variants. Lines connect reported means; no uncertainty band is drawn where standard deviations were not supplied.

Table 13: Adaptive candidate-specific versus uniform candidate cardinality control.

Method	Support F1	Length MAPE	Coverage	$\mathrm{nnz}(\mathbf{H})$	R_{Jac}	R_{spec}
Count + uniform length	0.9359 ± 0.0048	0.1061 ± 0.0059	0.8416 ± 0.0266	846.0 ± 29.6	0.00443	0.000076
Dual-control	0.9726 ± 0.0084	0.0483 ± 0.0145	0.9310 ± 0.0272	936.8 ± 27.2	0.00463	0.000079

clear applicability condition: the relation and background similarity components must be sufficiently distinguishable.

We further replace a fraction ρ of similarity observations according to the opposite-component contamination process (Table 16). Even low contamination causes a marked decline in support recovery: Dual-control F1 falls from 0.9726 at $\rho = 0$ to 0.8141 at $\rho = 0.01$ and 0.2985 at $\rho = 0.10$. The count remains exact through $\rho = 0.01$, but coverage and length accuracy deteriorate before the count estimate collapses. At $\rho = 0.50$, Dual-control retains only 2.6 hyperedges, with support F1 0.0171 and spectral redundancy 0.3502. These results show that the controllers suppress redundancy but cannot compensate for severely corrupted posterior evidence.

E.6 Construction cost and scalability

Increasing R from 1 to 10 raises total construction time from 0.01467 ± 0.00024 s to 0.06035 ± 0.00052 s and peak allocated memory from 9.31 to 18.10 MB (Table 17). Candidate generation and exact deduplication account for most of this increase; the costs of count selection, length control, and incidence packing remain comparatively stable.

Table 14: Length stress test. Each setting has a target and realized mean length of 10, 25, 50, or 100.

Target	Realized	Variant	Length MAE	Length MAPE	Coverage	$\mathrm{nnz}(\mathbf{H})$	Above capacity
10	10	Count-only	0.150	0.0166	0.9990	202.6	20
10	10	Dual-control	0.540	0.0477	0.9520	191.6	0
25	25	Count-only	0.420	0.0199	0.9972	506.0	20
25	25	Dual-control	2.150	0.0616	0.9244	467.0	0
50	50	Count-only	0.610	0.0128	0.9966	1007.4	20
50	50	Dual-control	3.680	0.0483	0.9310	936.8	0
100	100	Count-only	0.970	0.0111	0.9970	2013.8	20
100	100	Dual-control	5.900	0.0403	0.9452	1902.4	0

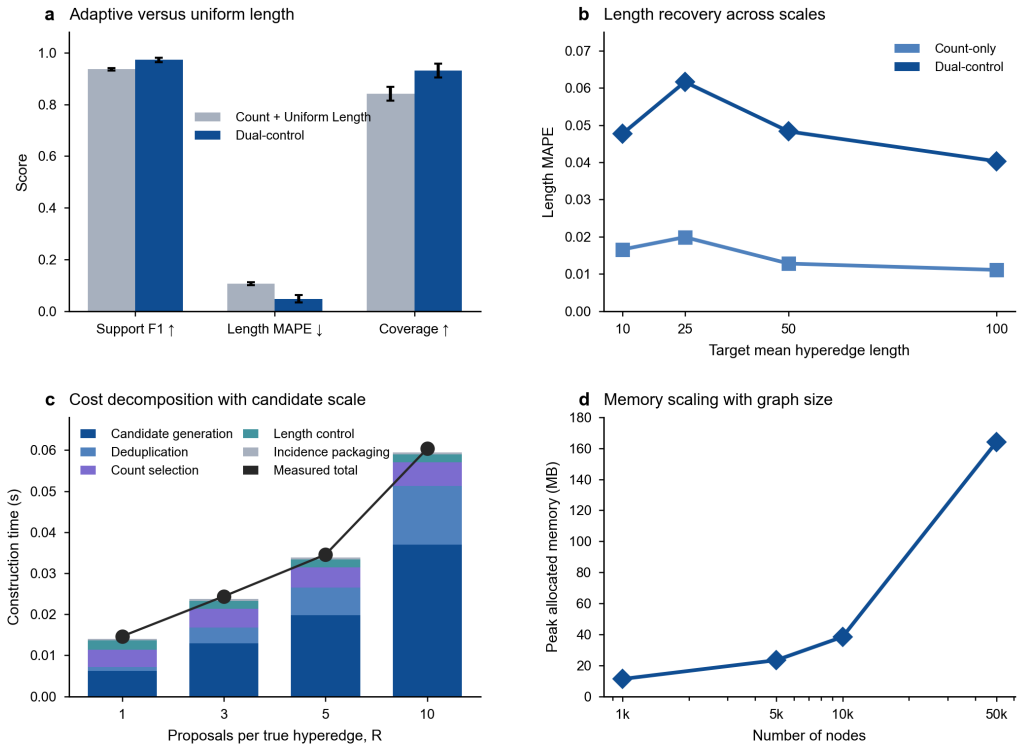

Figure 6: Length control and computational cost. **a**, Adaptive and uniform length allocation in the reference setting; error bars show the supplied standard deviations. **b**, length MAPE across target mean lengths. **c**, construction time decomposed by stage as R increases; the line shows total time. **d**, peak allocated GPU memory as the number of nodes increases.

With the candidate budget fixed, peak allocated memory grows from 11.25 MB at $N = 1{,}000$ to 164.04 MB at $N = 50{,}000$, while the final count and support F1 remain stable (Table 18). The measured runtime is nearly flat across these sizes. Because these short GPU measurements are dominated by fixed overhead, they should not be interpreted as constant asymptotic time; the memory trend provides the clearer scaling signal.

Overall, the synthetic results support three bounded conclusions. The joint spectral-and-coverage count controller recovers a compact set of representative supports when redundant proposals are present. Adaptive capacities outperform a uniform cap and enforce the intended structural bound. Both mechanisms remain dependent on the quality of posterior candidate evidence, with clear failure under weak component separation or substantial contamination.

Table 15: Effect of Gaussian-component separation D. Entries report Count-only/Dual-control.

D	Support F1	Length MAPE	R_{Jac}
2	0.3568/0.3505	0.6867/0.6988	0.3121/0.3119
4	0.8819/0.8558	0.0889/0.1410	0.0178/0.0161
6	0.9927/0.9726	0.0128/0.0483	0.0067/0.0046
8	0.9999/0.9969	0.0002/0.0059	0/0

Table 16: Dual-control under similarity contamination. Count error is the absolute error relative to $M^\star = 20$.

ρ	Support F1	Final M	Count error	Length MAPE	Coverage	$R_{\mathrm{Jac}}/R_{\mathrm{spec}}$
0	0.9726	20.0	0.0	0.0483	0.9310	0.0046/0.00010
0.01	0.8141	20.0	0.0	0.2049	0.8312	0.0289/0.00534
0.02	0.6969	19.8	0.2	0.2886	0.7746	0.0363/0.01011
0.05	0.4716	18.8	1.2	0.4169	0.7000	0.0498/0.01840
0.075	0.3686	18.0	2.0	0.4988	0.6680	0.0558/0.02302
0.10	0.2985	17.4	2.6	0.5561	0.6548	0.0558/0.0253
0.20	0.1685	17.4	2.6	0.6033	0.6486	0.0611/0.0287
0.30	0.1072	14.4	5.6	0.5436	0.6112	0.1288/0.0566
0.40	0.0381	5.4	14.6	0.9340	0.4414	0.3081/0.1405
0.50	0.0171	2.6	17.4	0.9801	0.2122	0.7433/0.3502

Table 17: Profiled component time in seconds as proposal redundancy R increases. Columns are grouped by component for reporting and do not denote a different execution order from the main method.

R	Raw/dedup.	Candidate	Dedup.	Count	Length	Pack	Total
1	20/20.0	0.00620	0.00094	0.00420	0.00231	0.00038	0.01467 ± 0.00024
3	60/37.2	0.01292	0.00381	0.00461	0.00198	0.00038	0.02440 ± 0.00023
5	100/49.8	0.01975	0.00675	0.00498	0.00192	0.00039	0.03456 ± 0.00049
10	200/74.4	0.03692	0.01432	0.00580	0.00194	0.00041	0.06035 ± 0.00052

Table 18: Node-scale stress test. Runtime statistics use 60 repetitions.

N	Mean time (s)	Median [Q1, Q3] (s)	Peak memory	Final M	Support F1
1,000	2.1344 ± 0.0426	2.1278 [2.1258, 2.1326]	11.25 MB	20.00	0.9727
5,000	2.1282 ± 0.0108	2.1313 [2.1246, 2.1331]	23.37 MB	20.00	0.9798
10,000	2.1258 ± 0.0043	2.1268 [2.1224, 2.1287]	38.42 MB	20.00	0.9798
50,000	2.1126 ± 0.0174	2.1239 [2.0883, 2.1246]	164.04 MB	19.67	0.9641