EDiS: Edge Disjoint Subgraph Sparsification Framework for Graph Neural Networks

Anonymous authors
Paper under double-blind review

Abstract

Sparse GNN training reduces computation, but deciding which edges to keep can be costly. Reusing one sparse graph is cheap but locks training to a fixed topology, while varying it across epochs can require repeated sampling or recomputation. We introduce `EDiS` (**E**dge-**Di**sjoint **S**ubgraph sparsification framework), which separates *one-time structural extraction* from *per-epoch graph composition*. `EDiS` decomposes the graph once into cacheable edge-disjoint subgraphs, then recombines them into graphs with edge-budget constraints across epochs and retention ratios without re-extracting structure.

Our default construction uses feature-based scores and successive maximum-score covering forests, while the same composition mechanism also supports alternative edge-selection rules. We provide a combinatorial analysis of the per-epoch sampler, the composition step that draws a training graph from the cached decomposition. We show that the stored decomposition deterministically preserves the graph's cut structure, and that any training graph composed from it, regardless of the edge-selection rule used, retains a guaranteed share of that structure. Across 19 homophilic, heterophilic, and large-scale node-classification benchmarks against 17 baselines under the same edge-budget, EDiS ranks first in aggregate accuracy, average rank, and gap to the best method. Ablations show the clearest benefits of structural decomposition and epoch variation at tight edge-budgets. Code is available at `https://anonymous.4open.science/r/EDSparse-9FB0/README.md`.

1 Introduction

Graph neural networks (GNNs) propagate information along the edges of the graph (Kipf & Welling, 2017), which determines the amount of information computed, exchanged, and stored at each epoch during training of the GNN. Sparsifying the graph, by selectively removing edges, can therefore reduce the computational costs (Chen et al., 2021), but adds a different cost of deciding *which* edges to keep (or remove). In some sampling-based methods, this cost dominates training itself. For example, on Reddit, a graph with 233K nodes and 11.6M edges, the sampling-based method AS-GCN (Huang et al., 2018) spends 20.7 seconds per epoch for sampling neighbors, versus 3.5 seconds for the forward and backward pass (Zeng et al., 2020). Thus, a sparsified graph does not guarantee economical training if the graph must be rebuilt frequently.

One option is to build a sparse graph once and reuse it every epoch to amortize the cost of sparsification. However, this also locks the GNN into the same set of edges for the entire training run (Chen et al., 2021; Seo et al., 2024). Another option is to change the sparse graph over time using different approaches such as repeated subgraph sampling in GraphSAINT (Zeng et al., 2020), incrementally grown spanning subgraph in SpanGNN (Gu et al., 2024), or computing a fresh random spanning tree at each epoch in GERN (Bonchi et al., 2025). While these methods show that changing sparse views during training helps, they closely couple a specific sampling or structural update method to achieve a fixed sparsity level. None of the current methods formulate the problem as a persistent, edge-disjoint structural library that can be built once and quickly recombined into different sparse graphs during training. This leads to an important question: *Can we extract graph structure once and reuse it to cheaply generate many different training graphs with specific edge-budget constraints?*

Addressing this question is the central idea behind `EDiS` (**E**dge-**Di**sjoint **S**ubgraph sparsification framework). `EDiS` turns edge-disjointness from a one-off computational step into a *cacheable*

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.

training paradigm. It decomposes the original graph into persistent and non-overlapping structural subgraphs only once. It then repeatedly recombines them into training graphs with edge-budget constraints across epochs and retention ratios, without recomputing the decomposition. In our default setup, subgraphs are successive connectivity-preserving sparse backbones, extracted from residual edges and cached before the GNN training begins.

While GNN architecture, parameters, and training objectives remain the same, the sparse message-passing graph changes across epochs. For example, a 30% edge budget does not mean the model is stuck with the same 30% of edges for the entire run, but that it processes 30% of edges per epoch, while different epochs are exposed to different parts of the original graph (Figure 1).

EDiS has three main components that can be combined together, *edge scoring*, *structural extraction*, and *budgeted composition*. While the first two are a one-time computation, budgeted composition is repeated at every epoch with specific edge-budgets. Edge scores rank connections, a residual selector groups them into cacheable subgraphs, and composition selects a training graph from those subgraphs. The subgraphs in extraction are disjoint, but edges can be repeated over time in the composed graphs ($\mathcal{G}_B^{(t)}$).

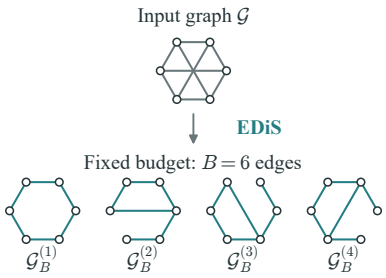


Figure 1: **EDiS generates sparse training graphs under a fixed edge budget.**

Our default algorithm pairs feature-based cosine scores with successive maximum-score covering forests. The first forest provides a high-score, connectivity-preserving backbone of the original graph. Once these edges are removed from the original graph, later forests span the connected components of the residual graph using previously unused edges. During training, the stored subgraphs are sampled with score-dependent priority and combined until the target budget is reached. Excess edges are trimmed, and any shortfall is filled from the residual pool (detailed in Section 3).

EDiS is not tied to one scoring or selection function. Edge scoring and residual selection functions are modular, so any combination shares the same edge-budget-constrained procedure (Section 2). Separating extraction from composition also clarifies what we can guarantee. For maximum-score residual forests, stored prefixes preserve score-threshold cut certificates. A single epoch's graph may omit or trim these forests, so we separately characterize conditional edge retention and fixed-cut survival under composition for any residual selector. Matched-budget ablations provide empirical evidence that structural grouping and epoch resampling contribute to predictive performance (Section 4.3).

We evaluate EDiS against 17 baselines on 19 datasets spanning homophilic, heterophilic, and large-scale graphs, separating sparse-graph inference using EDiS-Lite from full-graph inference using EDiS-Full, while isolating structural decomposition, epoch resampling, selector choice, and edge scoring through matched-edge-budget ablations. At 30% retention, EDiS-Lite achieves the highest average accuracy among ranked methods with an average rank of 3.32 and gap of 2.51 points, while EDiS-Full serves as a reference (Section 4). Cacheable subgraphs and graph variation give the largest gains at tight edge-budgets, while fixed graphs or simpler sampling remain competitive elsewhere.

Our contributions are as follows.

- **A cacheable, edge-budget-agnostic graph decomposition:** EDiS extracts structural subgraphs once, then recomposes them into training graphs with edge-budget constraints across epochs and sparsity levels, without rerunning extraction or changing the GNN architecture or loss.
- **A novel, selector-agnostic composition procedure:** To the best of our knowledge, EDiS is the first method to separate extraction from composition. Composition is independent of the extraction methods that build the subgraphs (e.g., maximum spanning forests or random selection).
- **A combinatorial analysis of the sampler:** For maximum-score residual forests, we prove score-aware cut preservation for the stored decomposition. For any residual selector, we separately characterize conditional retention and high-score cut survival in the composed graph.
- **A controlled empirical study of structural reuse:** Matched-edge-budget experiments across 17 baselines and 19 datasets test whether a cacheable structural decomposition and epoch variation justify their construction cost.

2 Related Work

Existing methods either decide which part of the graph to use every step, or decide once and fix the graph beforehand via sparsification. EDiS sits between these two settings, deciding once but still varying the graph every epoch. It builds a cacheable edge-disjoint structural library once, then composes an exact-budget graph at each epoch. This section relates EDiS to four lines of work.

Sampling-based GNN training. Neighbor sampling bounds neighbors per node (Hamilton et al., 2017). Layer-wise sampling instead scores whole node layers (Chen et al., 2018), and LADIES conditions each layer on the layer above it to stay connected (Zou et al., 2019). Subgraph methods sample subgraphs directly (Chiang et al., 2019) or correct the resulting bias (Zeng et al., 2020), and attribute-guided sampling prioritizes edges by similarity and diversity (Das et al., 2024). All of these resample the full topology at every step, rather than keeping an altered one. Other work avoids resampling by caching past embeddings (Fey et al., 2021), quantizing or compressing activations (Ding et al., 2021; Liu et al., 2022), repartitioning across workers (Wan et al., 2022), or compensating for missing cross-partition messages (Shi et al., 2023). EDiS instead organizes the topology once and reuses precomputed subgraphs under an explicit edge-budget, rather than resampling every step.

Graph sparsification. A large family of methods target edge removal before training begins, which includes degree- and similarity-based heuristics (Hamann et al., 2016; Satuluri et al., 2011; Liu et al., 2023), spectral and distance-preserving sparsifiers (Spielman & Srivastava, 2008; Peleg & Schäffer, 1989), and cut sparsifiers built by repeatedly extracting a maximum spanning forest (Benczúr & Karger, 1996; Forster & de Vos, 2023). In contrast, some methods learn which edges to retain jointly with graph learning (Chen et al., 2021; Seo et al., 2024; Zhang et al., 2024; Das et al., 2026; Zhang et al., 2025). However, no single method effectively preserves multiple graph properties of interest well (Chen et al., 2023). We employ similar sparsification principles in such as degree-, neighbor/similarity-, spanner-, and forest-based residual selection in EDiS, and composition treats them identically regardless of methods used to built the library.

Stochastic topology variation. DropEdge (Rong et al., 2020) and dropout-induced subgraphs (Luo et al., 2025) vary the topology by removing edges independently at random at each epoch. Other methods instead learn edge predictors (Chen et al., 2025) or stabilize message passing under a changing structure (Lee et al., 2025). These structures overlap substantially across epochs and carry no edge-budget or structural guarantee. By contrast, EDiS's epoch-to-epoch variation comes from a controlled, cacheable partition rather than independent per-edge noise.

Cacheable edge-disjoint decomposition. A few methods train over a sequence of sparse structures, SpanGNN (Gu et al., 2024) on grown subgraphs and GERN (Bonchi et al., 2025) on random spanning trees, but neither is persistent. Prior edge-disjoint methods use disjoint structures as sampled or sequential objects, not as a persistent, budget-agnostic decomposition recomposed into exact-budget graphs across epochs and retention ratios. EDiS is the first approach to decouple one-time extraction from per-epoch composition, so it stays cached and reused throughout training. Appendix B (Table 4) compares these methods by budget, reusability, exposure, and disjointness. Appendix J adds quantitative results against SpanGNN, GERN, and AGS-GNN.

3 EDiS: A Structured Sparsification Framework

Sparse GNN training faces a tradeoff between cost and coverage. A sparse graph built once is cheap to reuse. But the GNN then learns from the same fixed edges every epoch, which caps how much of the graph it can ever see. Changing the sparse graph over time exposes the GNN to more of the original topology. Rebuilding it from scratch every epoch, however, can make sparsification as expensive as the training it was meant to speed up.

EDiS separates these two concerns. *Decomposition* names the one-time step that builds the cached structural library. *Composition* names the per-epoch step that draws a graph from it. We extract structural information once, organize it into cacheable edge-disjoint subgraphs, and recombine them into an exact-budget graph each epoch. The GNN then processes only a fixed number of edges at a time, while seeing many structured subsets of the graph. The same GNN parameters are trained throughout, and only the sparse message-passing graph varies across epochs (Figure 2).

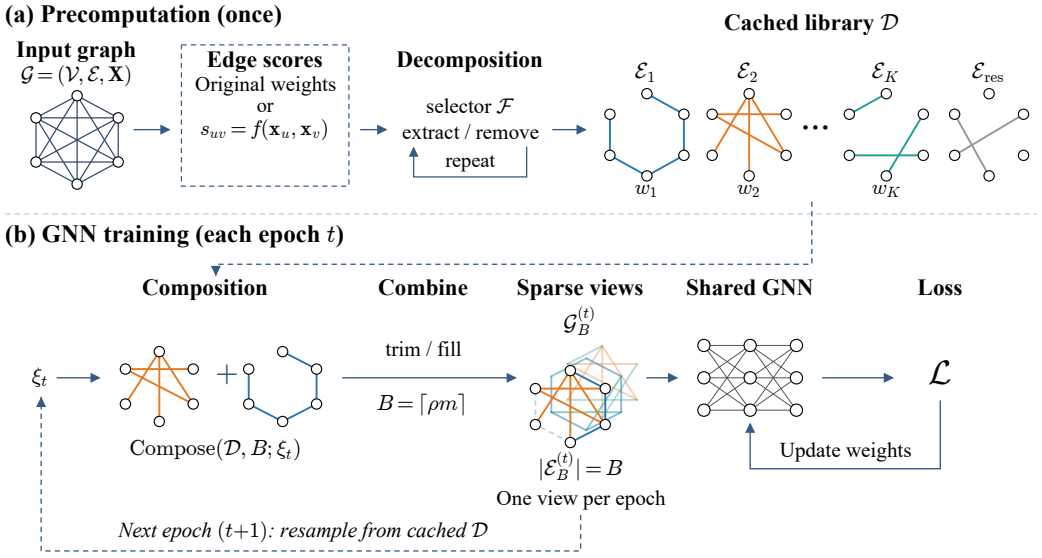


Figure 2: **Overview of the `EDiS` pipeline.**

3.1 One GNN, Many Sparse Views

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ be an undirected graph with n nodes, m edges, and node features $\mathbf{X}$. An edge-retention ratio $\rho \in (0, 1]$ sets the training budget $B = \lceil \rho m \rceil$, the number of edges used each epoch. `EDiS` *does not pick one B-edge graph and train on it for the whole run.* Instead, the same GNN trains on a sequence of different sparse graphs, one for each epoch t,

$$\mathcal{G}_B^{(t)} = \big(\mathcal{V}, \mathcal{E}_B^{(t)}, \mathbf{X}\big), \qquad |\mathcal{E}_B^{(t)}| = B, \tag{1}$$

where $\mathcal{E}_B^{(t)}$ is composed each epoch.

The GNN's parameters are shared across all epochs, and a GCN layer at epoch t still uses

$$\mathbf{H}_{\ell+1} = \sigma\Big(\widehat{\mathbf{A}}_B^{(t)} \mathbf{H}_\ell \mathbf{W}_\ell\Big), \qquad \mathbf{H}_0 = \mathbf{X},$$

where only the sparse adjacency $\widehat{\mathbf{A}}_B^{(t)}$ changes from epoch to epoch, while the learned weights $\mathbf{W}_\ell$ are updated through backpropagation. Over T epochs, the total number of distinct edges seen, $\big|\bigcup_{t=1}^{T} \mathcal{E}_B^{(t)}\big|$, can therefore be much larger than B, even though each epoch is limited by B.

The challenge is generating new sparse graphs without rerunning extraction each epoch. `EDiS` splits the method into two stages, a one-time decomposition (parametrized by edge score s and residual selector $\mathcal{F}$, defined next) and a cheap per-epoch composition.

$$\boxed{\mathcal{D} = \mathrm{Decompose}(\mathcal{G}; s, \mathcal{F}), \qquad \mathcal{E}_B^{(t)} = \mathrm{Compose}(\mathcal{D}, B; \xi_t).} \tag{2}$$

The decomposition $\mathcal{D}$ is computed once, and each epoch then needs only randomness ξ_t to compose a new B-edge graph from it. This composition is *not* adapted by the GNN loss, since the edge scores, subgraphs, and weights it draws on are all fixed before training begins.

3.2 One-Time Structural Decomposition

Each original edge $e = (u, v)$ gets a fixed score s_e, with higher scores marking edges the selector prefers. Our default score is a shifted cosine similarity between endpoint features,

$$s_{uv} = \frac{1}{2}\left(1 + \frac{\mathbf{x}_u^\top \mathbf{x}_v}{\|\mathbf{x}_u\|_2 \, \|\mathbf{x}_v\|_2}\right), \tag{3}$$

so this default depends on node features, not topology alone, though Section 4 evaluates other scores. Scores alone, however, do not decide the subgraphs. A *residual selector* $\mathcal{F}$ picks one subgraph at a time from whatever remains, mapping it to a subset at round k,

$$\mathcal{E}_k = \mathcal{F}\big(\mathcal{V}, \mathcal{E}_{\text{rem}}^{(k-1)}, s\big) \subseteq \mathcal{E}_{\text{rem}}^{(k-1)}. \tag{4}$$

The selector may use s or ignore it entirely, and need not return a forest or fixed edge count. EDiS treats any selector the same way during training, also supporting degree-based, neighbor-based, spanner, and random selectors (Appendix F.1).

Our default selector, $\mathcal{F}_{\max}$, groups edges into *structural subgraphs* rather than picking high-score edges one at a time. At each round, it extracts a maximum-score covering forest (**MaxCF**) from the edges still remaining. At round k,

$$\mathcal{E}_k = \mathcal{F}_{\max}\big(\mathcal{V}, \mathcal{E}_{\text{rem}}^{(k-1)}, s\big) \in \underset{F \text{ covering forest}}{\arg\max} \sum_{e \in F} s_e, \qquad \mathcal{E}_{\text{rem}}^{(k)} = \mathcal{E}_{\text{rem}}^{(k-1)} \setminus \mathcal{E}_k. \tag{5}$$

Extraction stops once a round limit is reached, a coverage target is met, or no edges remain, giving K subgraphs. These choices are fixed once, independent of B. That independence, not the absence of choices, is what *budget-agnostic* means here. The first subgraph $\mathcal{E}_1$ is thus a high-score, connectivity-preserving backbone. After removing those edges, $\mathcal{E}_2$ gives an alternative backbone from unused edges. The result is an edge-disjoint collection,

$$\mathcal{E} = \mathcal{E}_1 \sqcup \cdots \sqcup \mathcal{E}_K \sqcup \mathcal{E}_{\text{res}}, \tag{6}$$

where $\mathcal{E}_{\text{res}}$ holds any edges not assigned to a stored subgraph. Each subgraph also receives a weight,

$$w_k = \max\Big\{\textstyle\sum_{e \in \mathcal{E}_k} s_e,\ \varepsilon\Big\}, \tag{7}$$

so w_k is the subgraph's total score, floored at a small $\varepsilon > 0$ to keep sampling weights strictly positive. This floor is not a maximum over individual edges, so w_k always reflects the subgraph's aggregate score, not that of any single edge. The subgraphs, weights, and residual pool are computed once and stay fixed.

3.3 An Exact-Budget Graph at Every Epoch

At epoch t, EDiS decides which subgraphs to include, and in what order, by drawing a weighted random ordering without replacement. If R_r is the set of subgraphs still available before draw r,

$$\pi^{(t)} = (k_1, \ldots, k_K), \qquad \Pr(k_r = k \mid R_r) = \frac{w_k}{\sum_{j \in R_r} w_j}. \tag{8}$$

Higher-weight subgraphs are thus more likely to appear early, but no fixed order is imposed. We weight by total score, not average score. We add subgraphs one at a time in this order, stopping once the edge count reaches or passes B. If this overshoots B, we remove edges starting with the lowest-weight subgraph added. Each low-weight subgraph is removed in full until the remaining excess is smaller than the next subgraph, which is then trimmed by that amount and kept. At most one subgraph ends up partially trimmed. Every higher-weight subgraph stays whole. If instead there is a shortfall, it is filled uniformly from $\mathcal{E}_{\text{res}}$. Either way, the result satisfies

$$\boxed{|\mathcal{E}_B^{(t)}| = B.} \tag{9}$$

A deterministic alternative that always took the top subgraphs by expectation would instead repeat the same subgraphs every epoch, collapsing the epoch-to-epoch diversity that drives most of EDiS's gain at tight budgets (Appendix E.6, Appendix I.8). Exact construction and boundary cases appear in Appendix C.5. A worked example in Appendix D shows different selectors producing different subgraphs that still compose into the same edge budget. At small budgets, one high-weight subgraph may already fill B, so variation comes mainly from trimming within that subgraph, not subgraph choice. As B grows and several subgraphs are needed, epoch-to-epoch variation becomes increasingly subgraph-level.

This distinction from a fixed sparsifier matters. A one-shot method trains the GNN on the same edges every time,

$$\mathcal{E}_B, \mathcal{E}_B, \ldots, \mathcal{E}_B, \tag{10}$$

whereas EDiS trains the same GNN on a different composed graph each epoch,

$$\mathcal{E}_B^{(1)}, \mathcal{E}_B^{(2)}, \ldots, \mathcal{E}_B^{(T)}. \tag{11}$$

EDiS thus pairs sparse per-epoch computation with exposure to many structured views of the graph. Per-epoch composition can scale with B and K, not m, so the decomposition amortizes across epochs and budgets (Appendix F.2).

3.4 Structural Preservation for Decomposition and Composition

Guarantees for the cached decomposition $\mathcal{D}$ need not transfer to the composed graph $\mathcal{E}_B^{(t)}$ that the GNN actually sees each epoch, since composition can omit or trim subgraphs for a smaller budget. We therefore analyze the two objects separately.

Both guarantees below concern cuts carrying many high-score edges, so we index by a score threshold τ. Let $U_k = \mathcal{E}_1 \cup \cdots \cup \mathcal{E}_k$, and let $\mathcal{E}_{\geq\tau} = \{e \in \mathcal{E} : s_e \geq \tau\}$, $\mathcal{G}_{\geq\tau} = (\mathcal{V}, \mathcal{E}_{\geq\tau})$, and $U_{k,\geq\tau} = U_k \cap \mathcal{E}_{\geq\tau}$. For a nontrivial node set $S \subsetneq \mathcal{V}$ and graph H, $\delta_H(S)$ denotes the edges of H crossing the cut induced by S.

The stored decomposition. MaxCFs satisfy a bottleneck property (Fact 1). Together with the successive-forest cut certificate (Fact 2), this yields the guarantee below.

Theorem 1 (Score-threshold cut preservation)**.** *If each subgraph is a MaxCF of the current residual graph, then for every completed round $1 \leq k \leq K$, threshold τ, and nontrivial cut S,*

$$\left|\delta_{U_{k,\geq\tau}}(S)\right| \geq \min\left\{k,\ \left|\delta_{\mathcal{G}_{\geq\tau}}(S)\right|\right\}. \tag{12}$$

(Proof and the top-k corollary in Appendices E.2–E.3.) Informally, cuts that mattered in $\mathcal{G}$ keep crossing edges once enough subgraphs are kept. This holds for the default covering-forest selector.

The composed training graph. Once $B < |U_K|$, the sparse-budget regime this paper targets, $\mathcal{E}_B^{(t)}$ need not contain any U_k. We therefore bound retention for the composed graph directly, conditioning on the sampled subgraph order $\pi^{(t)}$. Let θ_k be subgraph k's retained fraction after trimming and $x_k(S,\tau)$ its edges crossing S with score at least τ.

Theorem 2 (Conditional retention under composition)**.**

$$\Pr\left(\mathcal{E}_B^{(t)} \cap \delta_{\mathcal{G}}(S) \cap \mathcal{E}_{\geq\tau} \neq \varnothing \;\middle|\; \pi^{(t)}\right) \geq 1 - \exp\left(-\sum_{k=1}^{K} \theta_k\, x_k(S,\tau)\right). \tag{13}$$

(Proof and the expected-retention identity in Appendix E.5.) Informally, a cut with enough high-score edges is unlikely to be fully severed, so the message-passing channel across it usually survives composition too, though no single subgraph's survival is guaranteed. Unlike Theorem 1, this bound is selector-agnostic, needing only a uniform trim within each subgraph, not forest structure, but it gives no guarantee that $\mathcal{G}_B^{(t)}$ stays connected at any fixed B (Appendix E.5).

Together, Theorems 1 and 2 deliberately decouple two guarantees typically bundled together, each scoped to where it holds. The stored decomposition carries a strong, selector-specific certificate established once at extraction time. The composed graph instead carries a selector-agnostic bound that holds for any selector, trading certificate strength for that generality, which is exactly what lets `EDiS` swap selectors freely.

4 Experimental Results

4.1 Experimental Setup

We evaluate on 19 node-classification benchmarks (six heterophilic, eight homophilic, and five large-scale), following Luo et al. (2024) with the Reddit dataset added, using accuracy except for Minesweeper, Questions, and ogbn-proteins (ROC-AUC). Full dataset characteristics are given in Table 6 (Appendix G). `EDiS` is compared against 17 methods. These are three reference anchors (a graph-free MLP, a spanning tree, a tuned full-graph GCN), 11 topology-guided sparsifiers, and three semantic-guided sparsifiers (Appendix G).

Our default pairs a GCN backbone with the MaxCF selector and cosine edge scores (Theorem 1). **EDiS-Lite** and **EDiS-Full** share this pipeline and differ only at inference, using the sparse composed graph or the complete graph. **Full GCN** is a separately tuned complete-graph reference, distinct from `EDiS` at 100% retention. Every method shares architecture, optimizer, and epoch budget per dataset (Appendix G.4). Full-batch training recomputes normalized adjacency from the current epoch graph, with scores guiding selection only, never propagation (Appendix G.3). Predictive runs, matched in count across methods, are averaged per dataset.

	Method	Cora	CiteSeer	Amazon Photo	WikiCS	Amazon Comp.	Coauthor CS	PubMed	Chameleon	Squirrel	Minesweeper
Ref.	MLP (No graph)	59.64 ±1.31	57.12 ±0.55	92.09 ±0.51	73.48 ±0.25	87.04 ±0.36	95.10 ±0.11	70.86 ±1.44	36.48 ±3.09	40.07 ±1.40	51.49 ±0.86
Ref.	Spanning Tree	80.10 ±0.26	69.42 ±0.56	93.97 ±0.08	76.65 ±0.22	90.45 ±0.25	95.47 ±0.19	79.48 ±0.46	36.73 ±4.52	37.59 ±1.74	67.03 ±0.58
Ref.	Full GCN	84.56 ±0.63	72.30 ±0.47	95.95 ±0.33	80.27 ±0.44	93.80 ±0.20	95.72 ±0.04	81.22 ±1.11	45.46 ±4.42	44.77 ±2.36	97.18 ±0.45
Ref.	**EDiS-Full (Ours)**	84.08 ±0.83	70.88 ±0.78	96.14 ±0.19	80.39 ±0.29	93.59 ±0.05	95.98 ±0.15	80.20 ±0.46	43.13 ±2.53	43.26 ±2.43	82.62 ±1.09
Topology-guided	Random	73.40 ±0.51	65.86 ±0.95	94.38 ±0.16	77.04 ±0.63	91.05 ±0.15	94.71 ±0.04	74.26 ±0.62	37.85 ±3.35	40.90 ±1.70	72.39 ±1.04
Topology-guided	Rank Deg.	76.34 ±0.78	65.10 ±0.24	93.79 ±0.05	76.13 ±0.53	88.04 ±0.28	94.68 ±0.06	76.38 ±0.46	40.35 ±3.82	40.17 ±1.89	71.09 ±0.21
Topology-guided	Local Deg.	73.52 ±0.47	64.76 ±0.84	94.49 ±0.03	75.69 ±0.48	89.28 ±0.33	94.93 ±0.22	77.32 ±1.08	41.37 ±2.59	41.80 ±1.66	72.53 ±1.15
Topology-guided	Forest Fire	71.98 ±0.39	63.82 ±1.05	95.40 ±0.22	79.20 ±0.58	92.57 ±0.12	94.67 ±0.33	73.46 ±0.85	41.28 ±1.96	43.15 ±1.90	70.76 ±0.29
Topology-guided	G-Spar	72.02 ±1.05	64.28 ±0.57	93.42 ±0.37	74.91 ±0.39	91.02 ±0.21	94.78 ±0.08	73.48 ±0.90	38.62 ±4.13	39.77 ±1.57	**78.97** ±0.51
Topology-guided	L-Sim	75.18 ±0.72	66.06 ±0.42	94.88 ±0.13	78.71 ±0.96	93.04 ±0.07	95.02 ±0.29	76.98 ±0.74	39.71 ±3.97	39.23 ±2.38	77.17 ±0.41
Topology-guided	SCAN	71.38 ±0.67	64.54 ±0.37	94.01 ±0.22	75.08 ±0.44	91.60 ±0.18	95.09 ±0.02	74.32 ±0.50	36.05 ±3.41	39.50 ±1.10	78.53 ±0.35
Topology-guided	L-Spar	71.38 ±1.48	64.32 ±0.57	94.07 ±0.20	75.10 ±0.25	90.95 ±0.22	95.03 ±0.22	73.48 ±0.90	36.11 ±3.06	39.17 ±1.31	78.45 ±0.86
Topology-guided	Spectral	72.86 ±0.85	**66.82** ±0.44	95.24 ±0.29	78.80 ±0.47	92.13 ±0.11	94.68 ±0.13	75.04 ±1.43	42.98 ±3.60	**43.94** ±1.18	70.48 ±1.94
Topology-guided	t-Spanner	72.60 ±0.90	65.70 ±0.63	94.95 ±0.24	78.07 ±0.23	91.51 ±0.11	94.64 ±0.27	74.74 ±0.59	38.71 ±3.72	42.65 ±2.01	71.13 ±1.03
Topology-guided	DSpar	73.90 ±0.36	64.70 ±1.22	**95.62** ±0.33	79.47 ±0.68	93.17 ±0.08	94.52 ±0.12	**77.86** ±0.46	39.73 ±3.87	43.48 ±1.56	72.56 ±2.00
Semantic guided	Unified-LTH	72.12 ±3.47	63.76 ±1.19	91.94 ±0.36	75.49 ±0.17	86.88 ±0.74	94.99 ±0.37	75.26 ±0.67	37.61 ±3.67	38.17 ±2.09	69.70 ±1.12
Semantic guided	AdaGLT	72.74 ±0.63	65.26 ±1.00	94.16 ±0.19	76.48 ±0.13	91.53 ±0.42	95.13 ±0.08	75.84 ±0.56	38.03 ±4.12	42.30 ±3.91	62.60 ±1.27
Semantic guided	MoG	73.66 ±0.77	65.28 ±1.50	95.45 ±0.29	77.48 ±0.66	92.55 ±0.05	94.94 ±0.20	71.56 ±0.54	43.40 ±3.13	43.25 ±1.12	74.02 ±0.32
	EDiS-Lite (Ours)	**77.16** ±0.98	66.08 ±1.23	95.60 ±0.15	**80.25** ±0.79	**93.35** ±0.21	**95.72** ±0.06	77.28 ±0.57	**43.75** ±4.14	42.63 ±2.04	73.82 ±0.97

	Method	Coauthor Phys.	Roman	Amazon Ratings	Questions	Proteins	Arxiv	Reddit	Pokec	Products	AM	GM
Ref.	MLP (No graph)	96.31 ±0.02	66.40 ±0.33	50.01 ±0.06	70.29 ±0.28	72.91 ±0.17	55.60 ±0.16	69.66 ±0.09	62.31 ±0.02	45.46 ±0.19	65.91	63.57
Ref.	Spanning Tree	96.90 ±0.02	82.45 ±0.56	49.95 ±0.77	73.89 ±0.61	74.13 ±0.16	67.27 ±0.12	86.07 ±0.14	69.36 ±0.01	67.79 ±0.09	73.41	71.05
Ref.	Full GCN	97.50 ±0.12	91.05 ±0.13	53.61 ±0.63	78.47 ±0.29	77.48 ±0.06	72.83 ±0.20	95.44 ±0.05	86.24 ±0.04	80.67 ±0.10	80.24	78.27
Ref.	**EDiS-Full (Ours)**	97.45 ±0.03	75.00 ±0.51	53.16 ±0.68	77.70 ±0.53	76.56 ±0.22	73.29 ±0.30	95.16 ±0.01	84.45 ±0.19	81.38 ±0.13	78.13	76.16
Topology-guided	Random	96.66 ±0.05	67.07 ±0.24	48.40 ±0.43	71.89 ±0.90	76.83 ±0.08	67.77 ±0.18	94.69 ±0.02	76.33 ±0.05	79.08 ±0.06	73.71	71.47
Topology-guided	Rank Deg.	96.40 ±0.17	70.53 ±0.33	51.23 ±0.52	71.49 ±0.48	66.01 ±0.78	62.59 ±0.07	77.28 ±0.04	67.29 ±0.10	63.13 ±0.02	70.95	69.05
Topology-guided	Local Deg.	96.30 ±0.18	70.79 ±0.31	51.47 ±0.71	74.39 ±0.83	71.10 ±0.00	68.34 ±0.17	90.82 ±0.15	80.07 ±0.09	77.62 ±0.16	74.03	72.11
Topology-guided	Forest Fire	96.91 ±0.03	67.43 ±0.62	48.37 ±0.22	71.34 ±0.46	**78.25** ±0.32	68.75 ±0.21	95.41 ±0.00	79.43 ±0.10	80.65 ±0.05	74.36	72.27
Topology-guided	G-Spar	96.73 ±0.06	72.20 ±0.42	49.64 ±0.61	70.35 ±1.77	70.39 ±0.56	65.86 ±0.06	90.05 ±0.01	67.35 ±0.01	72.55 ±0.05	72.44	70.31
Topology-guided	L-Sim	96.83 ±0.06	71.33 ±0.21	51.66 ±0.09	**75.39** ±0.60	77.95 ±0.11	70.69 ±0.08	94.35 ±0.05	76.56 ±0.06	79.20 ±0.09	75.26	73.06
Topology-guided	SCAN	96.84 ±0.03	73.37 ±0.23	50.25 ±0.61	72.72 ±0.45	70.62 ±0.42	67.36 ±0.37	90.28 ±0.01	68.56 ±0.02	72.56 ±0.15	72.77	70.50
Topology-guided	L-Spar	96.74 ±0.05	72.24 ±0.66	49.84 ±0.19	70.29 ±1.82	70.80 ±0.39	65.82 ±0.05	90.06 ±0.01	67.42 ±0.02	72.64 ±0.02	72.31	70.03
Topology-guided	Spectral	96.74 ±0.08	66.72 ±0.42	46.86 ±0.41	74.00 ±0.88	77.09 ±0.66	69.88 ±0.15	95.41 ±0.03	79.65 ±0.07	80.00 ±0.15	74.70	72.69
Topology-guided	t-Spanner	96.75 ±0.10	67.09 ±0.45	47.96 ±1.01	72.42 ±0.61	77.43 ±0.12	67.51 ±0.09	94.72 ±0.01	76.38 ±0.07	79.30 ±0.01	73.91	71.74
Topology-guided	DSpar	96.84 ±0.05	67.07 ±0.09	47.71 ±0.61	72.77 ±0.47	51.94 ±1.16	55.19 ±0.13	95.22 ±0.01	69.39 ±0.03	77.43 ±0.08	72.03	69.64
Semantic guided	Unified-LTH	96.49 ±0.00	65.43 ±0.58	40.49 ±0.52	72.42 ±0.64	67.15 ±0.00	56.37 ±0.00	92.68 ±0.00	68.23 ±0.00	75.26 ±0.00	70.55	68.02
Semantic guided	AdaGLT	96.65 ±0.00	**81.43** ±0.00	48.05 ±0.35	72.34 ±0.33	52.31 ±0.00	55.93 ±0.00	85.09 ±0.00	64.42 ±0.00	66.74 ±0.00	70.37	68.05
Semantic guided	MoG	96.93 ±0.10	70.44 ±0.60	51.24 ±0.33	68.90 ±1.68	72.72 ±1.75	56.42 ±0.16	90.13 ±0.08	66.94 ±0.07	68.27 ±0.05	72.29	70.39
	EDiS-Lite (Ours)	**97.06** ±0.09	68.58 ±0.12	**52.31** ±0.85	74.12 ±0.88	76.43 ±0.22	**71.41** ±0.26	**95.61** ±0.03	**81.19** ±0.08	**80.78** ±0.42	**75.95**	**74.01**

Table 1: Mean ± standard deviation (%) at retention ratio 0.30. Best, second-best, and third-best results are shaded blue/bold, gray/underlined, and tan, respectively. AM and GM denote the arithmetic and geometric means across all 19 datasets. The first four reference rows are excluded from ranking.

4.2 Predictive Performance

(RQ1, EDiS vs. 17 baselines) Does decoupling decomposition from composition cost accuracy at a matched budget? We benchmark it against all 17 baselines at $\rho = 0.30$ on 19 datasets, reporting the geometric mean (GM) and arithmetic mean (AM) across all 19 datasets while excluding reference anchors. Table 1 shows EDiS-Lite attaining the best GM and AM, ahead of every topology- and semantic-guided sparsifier at the same budget.

(RQ2) How large is the average margin? Table 2 aggregates results over all 19 datasets. Among the ranked methods, EDiS-Lite achieves the highest mean accuracy at both budgets. Since rank and gap are computed per dataset relative to the best ranked method and then averaged, EDiS-Lite attains average ranks of 3.32/4.05 and average gaps of only 2.51/1.59 percentage points at $\rho = 0.3/0.5$, respectively.

Table 2: **Summary of Table 1 over all 19 datasets** at $\rho = 0.3/0.5$. Rank and Gap are to the best ranked method per dataset, averaged; shading as in Table 1.

	Method	$\rho = 0.3$ Acc. ↑	$\rho = 0.3$ Rank ↓	$\rho = 0.3$ Gap ↓	$\rho = 0.5$ Acc. ↑	$\rho = 0.5$ Rank ↓	$\rho = 0.5$ Gap ↓
Topology-guided	Random	73.71	9.53	4.70	75.95	8.50	3.40
Topology-guided	Local Degree	74.03	7.61	4.29	76.60	8.18	2.75
Topology-guided	Rank Degree	70.95	11.26	7.54	73.36	10.87	5.99
Topology-guided	Forest Fire	74.36	7.34	4.15	77.10	5.11	2.25
Topology-guided	SCAN	72.77	8.97	5.43	75.71	9.03	3.64
Topology-guided	L-Spar	72.31	10.45	5.82	74.60	11.45	4.75
Topology-guided	G-Spar	72.44	10.55	5.81	74.49	11.87	4.86
Topology-guided	L-Sim	75.26	6.50	3.24	77.62	5.47	1.73
Topology-guided	Spectral	74.70	7.32	3.96	76.39	6.95	2.96
Topology-guided	t-Spanner	73.91	9.39	4.68	75.89	8.34	3.46
Topology-guided	DSpar	72.03	8.11	6.04	74.40	7.97	4.95
Sem guided	Unified-LTH	70.55	13.13	8.67	71.19	13.76	8.16
Sem guided	AdaGLT	70.37	11.37	8.76	71.76	12.32	7.59
Sem guided	MoG	72.29	9.42	6.03	74.02	9.58	5.33
	EDiS-Lite	**75.95**	**3.32**	**2.51**	**77.76**	**4.05**	**1.59**

(RQ3, EDiS-Lite vs. EDiS-Full) What does full-graph inference recover over sparse-graph inference, and what does it cost?

Both variants share the same trained model and differ only at inference. At tight budgets, EDiS-Lite can leave a sizeable gap to the independently tuned Full GCN reference, most visibly on Minesweeper and Pokec. Switching the evaluation graph to the full graph (EDiS-Full) recovers

much of it without retraining, and the gap shrinks as ρ grows (Figure 3). Reddit is a mild exception, where EDiS-Lite already sits close to Full GCN. That recovery is not free. Evaluation time rises and peak GPU memory increases mildly, while training cost is unchanged. These results make EDiS-Lite the default, with EDiS-Full reserved for cases where the accuracy gain is worth the extra inference cost. Full curves and the runtime and memory breakdown are in Appendices I.4 and H.

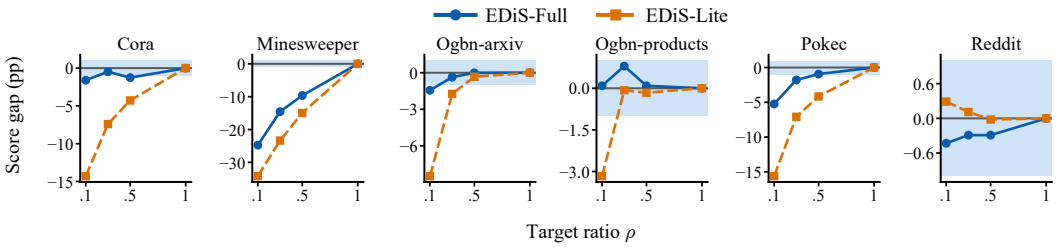


Figure 3: **Full-graph evaluation across regimes. MaxCF** with cosine scores.

4.3 Ablations and Generality

(RQ4, EDiS-Lite vs. matched-budget controls) Do cacheable subgraphs and per-epoch resampling drive the gains, or would any same-budget sampling do as well? We compare EDiS-Lite against three controls that fix the backbone, scores, and budget. **Frozen** reuses one sampled graph for the entire run, **Direct** samples edges by score without forming subgraphs, and **Shuffled groups** keeps EDiS's subgraph sizes but scrambles which edges belong to each subgraph, isolating structure from size.

Figure 4 shows EDiS-Lite matching or leading all three controls across the measured budget range, with the largest margin at tight budgets where each epoch graph holds few edges. Removing epoch variation (Frozen) hurts most under small budgets. Dropping subgraph formation (Direct) leaves a gap that closes only near the full graph. Shuffled groups, despite matching EDiS's subgraph sizes exactly, is consistently the weakest control. These results indicate that the gain comes from the structure of the decomposition and from resampling it every epoch, not merely from the number or size of edges available. Full results are given in Appendix I.3.

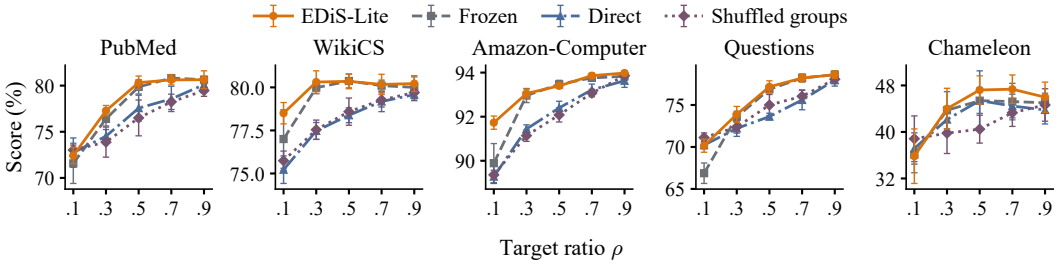


Figure 4: **Decomposition and resampling across edge budgets.**

(RQ5, selectors and edge scores) Does `EDiS` depend on which selector or edge score it uses, as the theory predicts it should not? Section 3.4 shows the retention bound governing the composed training graph holds for any residual selector, giving no inherent preference among them. We test this directly by sweeping seven selectors (MaxCF, MinCF, Degree, k-NN, Spanner, Random, Hybrid) and five edge scores (cosine, Euclidean, Dot, Uniform, MLP-score) through the same composition procedure with no other change.

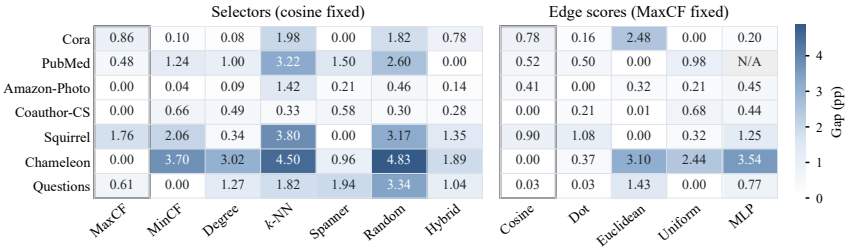


Figure 5: **Selector and score sensitivity.** GCN at $\rho = 0.3$. Cells show the gap to the best mean per row and panel (pp, lower is better). Outlines mark defaults. N/A marks PubMed–MLP-score, which is not applicable.

Fig. 5 confirms this. Across all seven selectors and five scores, most cells sit within a couple of points of the best choice on a given dataset, and no single selector or score dominates. The default MaxCF and cosine pairing stays competitive throughout. This independence extends to further datasets and to the GraphSAGE backbone (Appendix I.1–I.2, I.6).

(RQ6) How does extraction depth affect a fixed-budget graph? The cap $K_{\max}$ sets how many subgraphs we extract, and the budget B sets how many edges each epoch uses. A larger cap moves edges from the residual pool into subgraphs. We vary $K_{\max}$ on five large graphs at $\rho = 0.3$ (Figure 6). More subgraphs help on some datasets and hurt on others, and Lite and Full do not always agree. Thus, additional subgraphs do not always give a better graph, and once extraction stops before the cap, a larger $K_{\max}$ changes nothing (Appendix I.7).

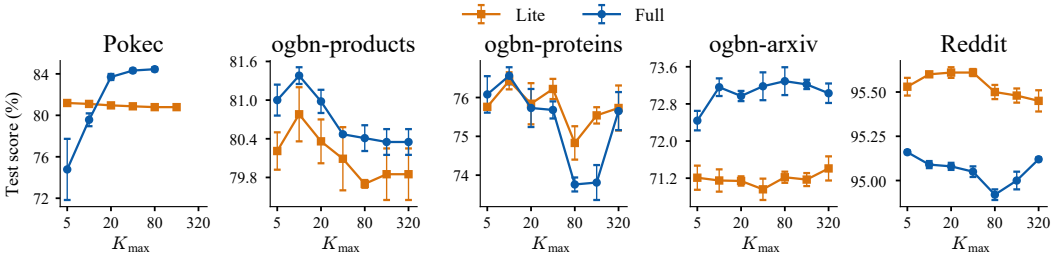


Figure 6: **Extraction depth at $\rho = 0.3$.** Reported means with standard-deviation bars. Proteins uses ROC-AUC, others accuracy. Each panel has its own scale.

(RQ7) Does the method work only with GCN? We fix MaxCF and cosine and switch the backbone from GCN to GraphSAGE (Figure 7). Both backbones improve with the budget and stay close, and neither is better overall. `EDiS` therefore works with either backbone (Appendix I.5).

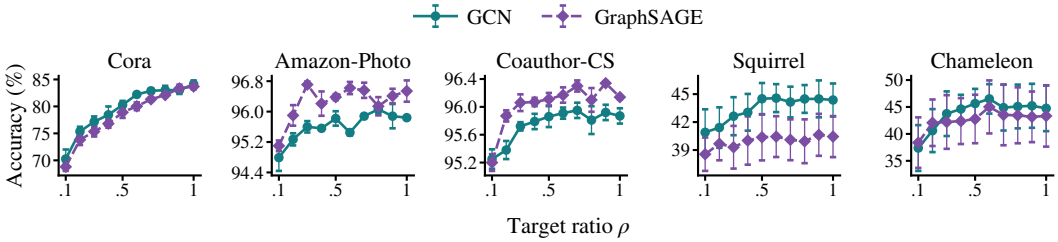


Figure 7: **Backbone comparison within EDiS-Lite.**

Further analysis. The appendix reports more experiments. It covers runtime and memory by budget (Appendix H), a cost analysis of construction and composition (Appendix F.2), the full ablation suite (Appendix I), full per-dataset results (Appendix G), and limitations (Appendix K).

5 Conclusion

We introduced `EDiS`, a framework that separates one-time structural extraction from per-epoch graph composition. We extract structure once, then compose many sparse training graphs from it, avoiding both a fixed topology and repeated extraction. The same cached library supplies exact-budget graphs at every epoch and retention level without changing the GNN architecture or objective. Our default MaxCF construction is just one instance of this framework, which stays modular across scores and selectors.

Across 19 node-classification benchmarks under the same edge-budget, EDiS-Lite achieves the highest mean accuracy among the ranked methods at both retention levels. EDiS-Full, reported as a reference using full-graph inference, further shows how much accuracy can be recovered when sparsification is restricted to training, at modest extra inference cost. Structural decomposition and epoch-to-epoch variation matter most at tight edge-budgets. More broadly, a cacheable structural representation that generates many sparse views can be more effective than searching for a single sparse graph to replace the original. Future work includes task-aware scores or subgraph weights within this decomposition, dynamic and temporal composition, and extensions to graph- and link-level prediction.

AI Use Disclosure

Generative AI tools were used to assist with polishing the manuscript and supporting research execution, including software implementation and experimental workflows. All AI-assisted content and code were reviewed, revised, and validated by the authors. The authors independently verified the reported results and claims and take full responsibility for the final content of this work.

Reproducibility Statement

We provide an anonymous code repository at https://anonymous.4open.science/r/EDSparse-9FB0/README.md containing the implementation of EDiS's decomposition and composition procedures. The appendix provides additional details on the datasets, baseline configurations, propagation and evaluation protocol, and hyperparameter settings, along with complete per-dataset numerical results, full proofs of all theorems, propositions, and lemmas, and the time-complexity analysis for EDiS and every selector family.

Ethics Statement

This work does not involve human subjects, personal data, or new data collection. All experiments use publicly available benchmark graphs under their original licenses. We are not aware of ethical risks specific to this methodology beyond those inherent to downstream applications of graph learning.

References

András A. Benczúr and David R. Karger. Approximating s-t Minimum Cuts in $\tilde{O}(n^2)$ Time. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pp. 47–55. ACM Press, 1996. doi: 10.1145/237814.237827.

Francesco Bonchi, Claudio Gentile, Francesco Paolo Nerini, André Panisson, and Fabio Vitale. Fast and Effective GNN Training through Sequences of Random Path Graphs. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1*, pp. 49–60. ACM, 2025. doi: 10.1145/3690624.3709301.

Jie Chen, Tengfei Ma, and Cao Xiao. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *6th International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rytstxWAW.

Tianlong Chen, Yongduo Sui, Xuxi Chen, Aston Zhang, and Zhangyang Wang. A Unified Lottery Ticket Hypothesis for Graph Neural Networks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 1695–1706. PMLR, 2021. URL https://proceedings.mlr.press/v139/chen21p.html.

Yuhan Chen, Haojie Ye, Sanketh Vedula, Alex Bronstein, Ronald Dreslinski, Trevor Mudge, and Nishil Talati. Demystifying Graph Sparsification Algorithms in Graph Properties Preservation. *Proceedings of the VLDB Endowment*, 17(3):427–440, 2023. doi: 10.14778/3632093.3632106.

Zhaoliang Chen, Zhihao Wu, Ylli Sadikaj, Claudia Plant, Hong-Ning Dai, Shiping Wang, Yiu-Ming Cheung, and Wenzhong Guo. ADEdgeDrop: Adversarial Edge Dropping for Robust Graph Neural Networks. *IEEE Transactions on Knowledge and Data Engineering*, 37(9):4948–4961, 2025. doi: 10.1109/TKDE.2025.3586369.

Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 257–266. ACM, 2019. doi: 10.1145/3292500.3330925.

Siddhartha Shankar Das, S M Ferdous, Mahantesh M. Halappanavar, Edoardo Serra, and Alex Pothen. AGS-GNN: Attribute-guided Sampling for Graph Neural Networks. In *Proceedings*

of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 538–549. ACM, 2024. doi: 10.1145/3637528.3671940.

Siddhartha Shankar Das, Naheed Anjum Arafat, Muftiqur Rahman, S M Ferdous, Alex Pothen, Mahantesh M. Halappanavar, and Danda B. Rawat. SGS-GNN: A Supervised Graph Sparsifier for Graph Neural Networks. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pp. 755–766. ACM, 2026. doi: 10.1145/3770855.3818110.

Mucong Ding, Kezhi Kong, Jingling Li, Chen Zhu, John Dickerson, Furong Huang, and Tom Goldstein. VQ-GNN: A Universal Framework to Scale up Graph Neural Networks using Vector Quantization. In *Advances in Neural Information Processing Systems*, volume 34, pp. 6733–6746. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/3569df159ec477451530c4455b2a9e86-Abstract.html.

Jack Edmonds. Matroid Partition. In George B. Dantzig and Arthur F. Veinott, Jr. (eds.), *Mathematics of the Decision Sciences: Part I*, volume 11 of *Lectures in Applied Mathematics*, pp. 335–345. American Mathematical Society, 1968.

Jack Edmonds. Matroids and the Greedy Algorithm. *Mathematical Programming*, 1(1):127–136, 1971. doi: 10.1007/BF01584082.

Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Jure Leskovec. GNNAutoScale: Scalable and Expressive Graph Neural Networks via Historical Embeddings. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 3294–3304. PMLR, 2021. URL https://proceedings.mlr.press/v139/fey21a.html.

Sebastian Forster and Tijn de Vos. Faster Cut Sparsification of Weighted Graphs. *Algorithmica*, 85 (4):929–964, 2023. doi: 10.1007/s00453-022-01053-4.

András Frank, Toshihide Ibaraki, and Hiroshi Nagamochi. On Sparse Subgraphs Preserving Connectivity Properties. *Journal of Graph Theory*, 17(3):275–281, 1993. doi: 10.1002/jgt.3190170302.

Yuan Gao, Rasmus Kyng, and Daniel A. Spielman. Robust and Practical Solution of Laplacian Equations by Approximate Elimination. arXiv:2303.00709, 2023. URL https://arxiv.org/abs/2303.00709.

Xizhi Gu, Hongzheng Li, Shihong Gao, Xinyan Zhang, Lei Chen, and Yingxia Shao. SpanGNN: Towards Memory-Efficient Graph Neural Networks via Spanning Subgraph Training. In *Machine Learning and Knowledge Discovery in Databases. Research Track*, volume 14943 of *Lecture Notes in Computer Science*, pp. 250–266. Springer, 2024. doi: 10.1007/978-3-031-70352-2_15.

Michael Hamann, Gerd Lindner, Henning Meyerhenke, Christian L. Staudt, and Dorothea Wagner. Structure-Preserving Sparsification Methods for Social Networks. *Social Network Analysis and Mining*, 6(1):22, 2016. doi: 10.1007/s13278-016-0332-2.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html.

Tanvir Hossain, Khaled Mohammed Saifuddin, Muhammad Ifte Khairul Islam, Farhan Tanvir, and Esra Akbas. Tackling Oversmoothing in GNN via Graph Sparsification. In *Machine Learning and Knowledge Discovery in Databases. Research Track and Demo Track*, volume 14948 of *Lecture Notes in Computer Science*, pp. 161–179. Springer, 2024. doi: 10.1007/978-3-031-70371-3_10.

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. In *Advances in Neural Information Processing Systems*, volume 33, pp. 22118–22133. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/fb60d411a5c5b72b2e7d3527cfc84fd0-Abstract.html.

Wenbing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. Adaptive Sampling Towards Fast Graph Representation Learning. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/hash/01eee509ee2f68dc6014898c309e86bf-Abstract.html.

Adarsh Jamadandi, Celia Rubio-Madrigal, and Rebekka Burkholz. Spectral Graph Pruning Against Over-Squashing and Over-Smoothing. In *Advances in Neural Information Processing Systems*, volume 37, pp. 10348–10379. Curran Associates, Inc., 2024. doi: 10.52202/079017-0331. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/140aac600566125915df7e74ff538f66-Abstract-Conference.html.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Joseph B. Kruskal, Jr. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956. doi: 10.1090/S0002-9939-1956-0078686-7.

Dooho Lee, Myeong Kong, Sagad Hamid, Cheonwoo Lee, and Jaemin Yoo. Aggregation Buffer: Revisiting DropEdge with a New Parameter Block. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 33181–33204. PMLR, 2025. URL https://proceedings.mlr.press/v267/lee25n.html.

Jure Leskovec and Christos Faloutsos. Sampling from Large Graphs. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 631–636. ACM, 2006. doi: 10.1145/1150402.1150479.

Langzhang Liang, Fanchen Bu, Zixing Song, Zenglin Xu, Shirui Pan, and Kijung Shin. Mitigating Over-Squashing in Graph Neural Networks by Spectrum-Preserving Sparsification. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 37051–37070. PMLR, 2025. URL https://proceedings.mlr.press/v267/liang25a.html.

Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser Nam Lim. Large Scale Learning on Non-Homophilous Graphs: New Benchmarks and Strong Simple Methods. In *Advances in Neural Information Processing Systems*, volume 34, pp. 20887–20902. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/ae816a80e4c1c56caa2eb4e1819cbb2f-Abstract.html.

Zirui Liu, Kaixiong Zhou, Fan Yang, Li Li, Rui Chen, and Xia Hu. EXACT: Scalable Graph Neural Networks Training via Extreme Activation Compression. In *10th International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=vkaMaq95_rX.

Zirui Liu, Kaixiong Zhou, Zhimeng Jiang, Li Li, Rui Chen, Soo-Hyun Choi, and Xia Hu. DSpar: An Embarrassingly Simple Strategy for Efficient GNN Training and Inference via Degree-Based Sparsification. *Transactions on Machine Learning Research*, 2023. URL https://openreview.net/forum?id=SaVEXFuozg.

Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Classic GNNs are Strong Baselines: Reassessing GNNs for Node Classification. In *Advances in Neural Information Processing Systems*, volume 37, pp. 97650–97669. Curran Associates, Inc., 2024. doi: 10.52202/079017-3098. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/b10ed15ff1aa864f1be3a75f1ffc021b-Abstract-Datasets_and_Benchmarks_Track.html. Datasets and Benchmarks Track.

Yuankai Luo, Xiao-Ming Wu, and Hao Zhu. Beyond Random Masking: When Dropout meets Graph Convolutional Networks. In *13th International Conference on Learning Representations*, pp. 11149–11168, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/1ddcd22e3a0f830db8ecd3fd242c2c56-Abstract-Conference.html.

Péter Mernyei and Cătălina Cangea. Wiki-CS: A Wikipedia-Based Benchmark for Graph Neural Networks. arXiv:2007.02901, 2020. URL https://arxiv.org/abs/2007.02901. Presented at the ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+).

Hiroshi Nagamochi and Toshihide Ibaraki. A Linear-Time Algorithm for Finding a Sparse k-Connected Spanning Subgraph of a k-Connected Graph. *Algorithmica*, 7(1-6):583–596, 1992. doi: 10.1007/BF01758778.

George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. An Analysis of Approximations for Maximizing Submodular Set Functions—I. *Mathematical Programming*, 14(1):265–294, 1978. doi: 10.1007/BF01588971.

David Peleg and Alejandro A. Schäffer. Graph Spanners. *Journal of Graph Theory*, 13(1):99–116, 1989. doi: 10.1002/jgt.3190130114.

Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A Critical Look at the Evaluation of GNNs under Heterophily: Are We Really Making Progress? In *11th International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=tJbbQfw-5wv.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *8th International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Hkx1qkrKPr.

Venu Satuluri, Srinivasan Parthasarathy, and Yiye Ruan. Local Graph Sparsification for Scalable Clustering. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, pp. 721–732. ACM, 2011. doi: 10.1145/1989323.1989399.

Hyunjin Seo, Jihun Yun, and Eunho Yang. TEDDY: Trimming Edges with Degree-based Discrimination Strategy. In *12th International Conference on Learning Representations*, pp. 33937–33959, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/90d1fc07f46e31387978b88e7e057a31-Abstract-Conference.html.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of Graph Neural Network Evaluation. arXiv:1811.05868, 2018. URL https://arxiv.org/abs/1811.05868. Presented at the NeurIPS 2018 Workshop on Relational Representation Learning (R2L).

Zhihao Shi, Xize Liang, and Jie Wang. LMC: Fast Training of GNNs via Subgraph Sampling with Provable Convergence. In *11th International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=5VBBA91N6n.

Daniel A. Spielman and Nikhil Srivastava. Graph Sparsification by Effective Resistances. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pp. 563–568. ACM, 2008. doi: 10.1145/1374376.1374456.

Elli Voudigari, Nikos Salamanos, Theodore Papageorgiou, and Emmanuel J. Yannakoudakis. Rank Degree: An Efficient Algorithm for Graph Sampling. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pp. 120–129. IEEE, 2016. doi: 10.1109/ASONAM.2016.7752223.

Cheng Wan, Youjie Li, Ang Li, Nam Sung Kim, and Yingyan Lin. BNS-GCN: Efficient Full-Graph Training of Graph Convolutional Networks with Partition-Parallelism and Random Boundary Node Sampling. In *Proceedings of Machine Learning and Systems*, volume 4, pp. 673–693, 2022. URL https://proceedings.mlsys.org/paper_files/paper/2022/hash/676638b91bc90529e09b22e58abb01d6-Abstract.html.

Xiaowei Xu, Nurcan Yuruk, Zhidan Feng, and Thomas A. J. Schweiger. SCAN: A Structural Clustering Algorithm for Networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 824–833. ACM, 2007. doi: 10.1145/1281192.1281280.

Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting Semi-Supervised Learning with Graph Embeddings. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 40–48. PMLR, 2016. URL https://proceedings.mlr.press/v48/yanga16.html.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graph-SAINT: Graph Sampling Based Inductive Learning Method. In *8th International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJe8pkHFwS.

Guibin Zhang, Kun Wang, Wei Huang, Yanwei Yue, Yang Wang, Roger Zimmermann, Aojun Zhou, Dawei Cheng, Jin Zeng, and Yuxuan Liang. Graph Lottery Ticket Automated. In *12th International Conference on Learning Representations*, pp. 31044–31068, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/83c230118e9f6688ba8f20bfef99e6da-Abstract-Conference.html.

Guibin Zhang, Xiangguo Sun, Yanwei Yue, Chonghe Jiang, Kun Wang, Tianlong Chen, and Shirui Pan. Graph Sparsification via Mixture of Graphs. In *13th International Conference on Learning Representations*, pp. 92735–92763, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/e7947b5e1d30864ebbe8714dbdd611d9-Abstract-Conference.html.

Shiqi Zhang, Renchi Yang, Jing Tang, Xiaokui Xiao, and Bo Tang. Efficient Approximation Algorithms for Spanning Centrality. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3386–3395. ACM, 2023. doi: 10.1145/3580305.3599323.

Difan Zou, Ziniu Hu, Yewen Wang, Song Jiang, Yizhou Sun, and Quanquan Gu. Layer-Dependent Importance Sampling for Training Deep and Large Graph Convolutional Networks. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/hash/91ba4a4478a66bee9812b0804b6f9d1b-Abstract.html.

Appendix Contents

A Notation **16**

B Extended Related Work **17**

C Formal Definitions and Construction **19**

C.1 Graph, GNN, and Edge Budget . . . 19

C.2 Edge Scores and Residual Selectors . . . 19

C.3 Maximum-Score Residual Forests . . . 19

C.4 Classical Forest Properties . . . 20

C.5 Exact Stochastic Budgeted Composition . . . 20

C.6 Structural Preservation of the Stored Decomposition . . . 20

C.7 Retention After Budgeted Composition . . . 21

D Worked Examples of Structural Selectors **22**

E Theoretical Understanding and Proofs **23**

E.1 Reuse and Exact-Budget Feasibility . . . 23

E.2 Score-Threshold Certificate . . . 23

E.3 Top-k Cut-Score Corollary . . . 24

E.4 Equivalent Interpretations . . . 24

E.5 Composition Retention . . . 25

E.6 Optional Composition Variants . . . 26

F Selectors and Computational Cost **27**

F.1 Family of Residual Selectors . . . 27

F.2 Cost of Construction and Epoch Composition . . . 28

G Experimental Details **28**

G.1 Dataset . . . 28

G.2 Baseline Settings . . . 29

G.3 Propagation and Evaluation Settings . . . 29

G.4 Hyperparameter Settings . . . 29

G.5 Complete numerical results . . . 30

H Runtime and Memory Details **34**

H.1 Lite versus Full Evaluation Trade-off . . . 34

H.2 End-to-End Runtime Against Baselines . . . 35

I Ablations and Additional Profiles **36**

I.1 Extended selector ablation . . . 36

I.2 Extended edge-score ablation . . . 37

I.3 Extended mechanism ablation . . . 38

I.4 Extended inference-graph ablation . . . 39

I.5 Extended backbone comparison . . . 40

I.6 Selector and score robustness under GraphSAGE . . . 41

I.7 Extraction depth on large graphs . . . 42

I.8 Coverage, overlap, and decomposition reuse . . . 43

J Additional Experiments **44**

J.1 Comparison with SpanGNN and GERN . . . 44

J.2 Comparison with MoG under GraphSAGE . . . 44

K Limitations **45**

A NOTATION

This appendix collects the notation used throughout EDiS, including the symbols introduced for the proofs in Appendix E. Table 3 lists the main symbols, grouped by where each is introduced. Symbols that appear only in a single appendix result are instead defined locally, at the point where they are used. The letter H is intentionally reused as a generic placeholder for the current graph.

Symbol	Meaning
Graph and edge-budget	
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$	Attributed undirected graph ($\mathcal{V}, \mathbf{X}$ fixed; only the edge set varies below)
$n = \lvert\mathcal{V}\rvert$, $m = \lvert\mathcal{E}\rvert$	Number of nodes and edges
ρ	Edge-retention ratio
$B = \lceil \rho m \rceil$	Number of active edges per epoch
One-time edge decomposition	
s_e	Score assigned to edge e
$\mathcal{F}$, $\mathcal{F}_k$	Residual selector ($\mathcal{F}_k$: round-dependent schedule)
$\mathcal{E}_{\text{rem}}^{(k)}$	Edges remaining after decomposition round k
$\mathcal{E}_k$	Subgraph extracted at round k
K	Number of extracted subgraphs
$K_{\max}$	Cap on extraction rounds ($K \le K_{\max}$)
$\mathcal{E}_{\text{res}}$	Residual edges after the final round
$U_k = \mathcal{E}_1 \cup \cdots \cup \mathcal{E}_k$	Union of the first k subgraphs (U_K: full stored union)
Per-epoch graph construction	
T	Total number of training epochs
Decompose	One-time decomposition of $\mathcal{G}$ under score s and selector $\mathcal{F}$ into the stored $\mathcal{D}$
Compose	Budgeted composition, mapping the stored decomposition and B to $\mathcal{E}_B^{(t)}$
$\mathcal{D}$	Edge partition with cached subgraph weights, reused across edge-budgets
w_k	Positive subgraph weight $\max\{\sum_{e\in\mathcal{E}_k} s_e, \varepsilon\}$
R_r, k_r	Remaining subgraph indices and the subgraph selected at draw r
ξ_t	Epoch randomness, including $\pi^{(t)}$ and within-pool draws
$A^{(t)}$	Subgraph indices selected at epoch t
$\widetilde{\mathcal{E}}_k^{(t)}$	Selected subgraph edges remaining after trimming
$Q_B^{(t)}$	Residual edges used to complete the per-epoch edge-budget
$\pi^{(t)}$	Weighted subgraph order for epoch t
$\mathcal{E}_B^{(t)}$	Active edge set at epoch t
$\mathcal{G}_B^{(t)}$	Graph for message passing at epoch t ($\mathcal{V}, \mathbf{X}$ unchanged)
$\widehat{\mathbf{A}}_B^{(t)}$	Normalized adjacency of $\mathcal{G}_B^{(t)}$ used for message passing
Score-aware guarantee	
τ	Edge-score threshold
$\mathcal{E}_{\ge\tau} = \{e \in \mathcal{E} : s_e \ge \tau\}$	Edges with score at least τ
$\mathcal{G}_{\ge\tau} = (\mathcal{V}, \mathcal{E}_{\ge\tau})$, $U_{k,\ge\tau} = U_k \cap \mathcal{E}_{\ge\tau}$	Threshold graph and the stored union restricted to it
$a_j(S)$, $b_j(S)$	j-th largest crossing score at cut S in $\mathcal{G}$ and in U_k
$T_k(S;A) = \sum_{j=1}^{\min\{k,\lvert\delta_A(S)\rvert\}} s_{(j)}(\delta_A(S))$	Sum of the $\min\{k,\cdot\}$ largest scores crossing S in A
$\delta_H(S)$	Edges of H crossing the cut $(S, \mathcal{V}\setminus S)$
$d_H(v)$	Degree of v in H
Proof-specific notation (Appendix E)	
s	Additive score function, extending s_e to edge sets
$\omega(F) = \sum_{e\in F} \omega_e$	Generic additive edge-weight function
D_k	Stored subgraph k, with $D_0 = \mathcal{E}_{\text{res}}$ and $D_k = \mathcal{E}_k$ for $k \ge 1$
$n_k = \lvert D_k\rvert$	Size of subgraph k
r_k	Number of edges retained from subgraph k
$\theta_k = r_k/n_k$	Retained fraction / inclusion probability given the order (0 if the pool is empty)
$x_k(S,\tau) = \lvert D_k \cap \delta_{\mathcal{G}}(S) \cap \mathcal{E}_{\ge\tau}\rvert$	Edges of subgraph k crossing S with score at least τ

Table 3: Main notation used in EDiS.

B Extended Related Work

Sampling-based GNN training. Many GNN methods reduce training cost by processing only part of the graph at a time rather than constructing a new topology. Neighbor sampling selects a limited number of neighbors per node at every GNN layer (Hamilton et al., 2017). Layer-wise sampling instead selects a fixed number of nodes by importance (Chen et al., 2018). LADIES extends this idea by picking each layer's nodes based on which nodes were kept in the layer above it, so the sampled nodes stay connected from one layer to the next (Zou et al., 2019). Other methods divide the graph into clusters and train on these smaller parts (Chiang et al., 2019), or sample training subgraphs and pass messages only within them (Zeng et al., 2020). AGS-GNN (Das et al., 2024) guides node and subgraph sampling by feature similarity and diversity, and reappears below as an edge-disjoint variant. Related approaches reuse historical node representations (Fey et al., 2021), quantize activations (Ding et al., 2021) or compress them further (Liu et al., 2022), or repartition the graph across workers (Wan et al., 2022) or compensate for missing cross-partition messages (Shi et al., 2023).

These methods reduce how much of the graph is processed at once, but the resulting structures are repeatedly resampled from the original topology. `EDiS` instead organizes the topology itself and controls which precomputed subgraphs are used under a specified edge-retention budget.

Graph sparsification and topology modification. A second group of methods changes which edges are used for message passing. Sparsification methods remove edges while preserving structural or task-relevant information. Classical topology-based approaches include uniform random removal, Local Degree (Hamann et al., 2016), Rank Degree (Voudigari et al., 2016), Forest Fire (Leskovec & Faloutsos, 2006), and SCAN (Xu et al., 2007). Similarity-based approaches include L-Spar, G-Spar, and L-Sim, which retain edges using neighborhood similarity (Satuluri et al., 2011). Other structural approaches include spectral sparsification via effective resistance (Spielman & Srivastava, 2008), t-spanners (Peleg & Schäffer, 1989), and degree-based DSpar (Liu et al., 2023). A classical line of work builds cut sparsifiers by repeatedly extracting a maximum spanning forest from what is left of the graph, which gives provable guarantees on how well cut sizes are preserved (Benczúr & Karger, 1996; Forster & de Vos, 2023). `EDiS` reuses this same repeated-forest-extraction idea, but keeps each extracted forest as a separate, cacheable training subgraph instead of merging them into one fixed sparsifier.

A separate line learns or adapts which elements to retain. UGS (Chen et al., 2021), from the Unified Lottery Ticket framework, prunes edges and model parameters together over repeated training rounds. TEDDY (Seo et al., 2024) instead removes edges in a single pass using a degree-based rule, avoiding UGS's repeated pruning cycles. AdaGLT (Zhang et al., 2024) adds adaptive, automated sparsification. SGS-GNN (Das et al., 2026) learns which edges to keep using supervision and lets the user directly set the target sparsity. MoG (Zhang et al., 2025) combines sparsification experts with different pruning criteria and sparsity levels into task-adaptive graphs.

Different sparsification methods preserve different graph properties, and no single method preserves all of them well (Chen et al., 2023). Removing edges therefore affects not only training cost but also how information propagates. Sparsification has also been used against oversmoothing and oversquashing, via spectrum-preserving pruning (Liang et al., 2025; Jamadandi et al., 2024) or a truss-based approach that removes edges outside dense subgraphs (Hossain et al., 2024). These methods generally focus on identifying or adapting a useful sparse topology for message passing, rather than on how that topology is exposed to the GNN across training steps.

A related line changes the topology stochastically during training. DropEdge removes a random subset of edges at each iteration (Rong et al., 2020). Even feature dropout, which does not explicitly remove edges, induces different stochastic subgraphs across feature dimensions. A theoretical analysis attributes part of dropout's ability to mitigate oversmoothing in GCNs to this variation (Luo et al., 2025). Other methods replace random removal with learned edge predictors (Chen et al., 2025), or add parameters to stabilize message passing under changing structures (Lee et al., 2025).

Structures generated by DropEdge or dropout-induced subgraphs at different training steps can substantially overlap in the edges they use. `EDiS` instead decomposes the graph once and reuses the resulting non-overlapping subgraphs, composing them under an explicit edge-retention budget that lets the message-passing topology vary while directly controlling the number of retained edges.

Cacheable edge-disjoint decomposition. A smaller body of work trains GNNs using multiple graph structures rather than one sparse topology. The AGS-GNN sampling framework (Das et al., 2024) introduced above also includes AGS-GS-Disjoint, which sequentially constructs sparse subgraphs with different heuristics, removing selected edges before constructing the next.

Our objective still differs, and so does the intent behind the construction itself. AGS-GS-Disjoint's edge-disjoint subgraphs feed a dual-channel sampler that uses submodular function maximization to select similar and diverse neighborhoods, adapting to homophily and heterophily. Its per-epoch sample draws a fixed number of these cached subgraphs padded with random residual edges, with no exact-edge-count guarantee or attached structural theory. `EDiS` instead studies edge-disjoint decomposition as a general sparsification primitive. Any residual selector composes into an exact-B-edge training graph, with a proven cut-preservation guarantee for the stored decomposition and a retention guarantee for the composed graph that holds regardless of selector. This lets us directly compare organizing a fixed number of retained edges into non-overlapping subgraphs against alternative ways of using the same budget, rather than optimizing for informative neighborhoods.

SpanGNN (Gu et al., 2024) trains using spanning subgraphs to reduce memory, and MoG (Zhang et al., 2025) combines sparsification experts to produce adaptive sparse graphs. GERN (Bonchi et al., 2025) also exposes a GNN to multiple sparse structures, progressively training over a sequence of random spanning trees transformed into path graphs. Unlike `EDiS`, it does not build a persistent edge-disjoint repository, progressing instead through independently generated sparse path structures toward efficient training rather than budget-controlled composition of a fixed repository. To our knowledge, none of them partitions the original edge set into a persistent, mutually edge-disjoint collection that is later recombined under a retained-edge constraint.

These approaches confirm that exposing a GNN to multiple sparse structures can help, though each still addresses a different question from `EDiS`, namely how to allocate a fixed retained-edge budget across a cacheable collection of non-overlapping subgraphs. The graph is decomposed once, unused edges remain explicitly accounted for in a residual subgraph, and the subgraphs are recomposed under the same edge-budget constraint at every epoch, so different training-graph constructions can be compared under the same retained-edge count. Table 4 compares these methods along these axes.

Method	Type	Edge budget	Reusable	Multi-struct.	Edge-disjoint
t-Spanner (Peleg & Schäffer, 1989)	Sparsification	✓	✓	✗	N/A
Forest Fire (Leskovec & Faloutsos, 2006)	Sampling	✗	✗	✗	N/A
SCAN (Xu et al., 2007)	Sparsification	✓	✓	✗	N/A
Spectral (Spielman & Srivastava, 2008)	Sparsification	✓	✓	✗	N/A
L-Spar (Satuluri et al., 2011)	Sparsification	✓	✓	✗	N/A
G-Spar (Satuluri et al., 2011)	Sparsification	✓	✓	✗	N/A
L-Sim (Chen et al., 2023)	Sparsification	✓	✓	✗	N/A
Local Degree (Hamann et al., 2016)	Sparsification	✓	✓	✗	N/A
Rank Degree (Voudigari et al., 2016)	Sampling	✗	✗	✗	N/A
Neighbor Sampling (Hamilton et al., 2017)	Sampling	✗	✗	✓	✗
FastGCN (Chen et al., 2018)	Sampling	✗	✗	✓	✗
LADIES (Zou et al., 2019)	Sampling	✗	✗	✓	✗
Cluster-GCN (Chiang et al., 2019)	Sampling	✗	✓	✓	✗
DropEdge (Rong et al., 2020)	Stochastic	✗	✗	✓	✗
GraphSAINT (Zeng et al., 2020)	Sampling	✗	✗	✓	✗
Unified-LTH / UGS (Chen et al., 2021)	Learned sparsif.	✓	✓	✗	N/A
Cut Sparsification (Forster & de Vos, 2023)	Sparsification	✓	✓	✗	N/A
DSpar (Liu et al., 2023)	Sparsification	✓	✓	✗	N/A
AGS-GS-Disjoint (Das et al., 2024)	Sampling	✗	✓	✓	✓
TEDDY (Seo et al., 2024)	Learned sparsif.	✓	✓	✗	N/A
AdaGLT (Zhang et al., 2024)	Learned sparsif.	✓	✓	✗	N/A
SpanGNN (Gu et al., 2024)	Multi-structure	✗	✓	✓	✗
Feature Dropout (Luo et al., 2025)	Stochastic	✗	✗	✓	✗
Learned Edge Predictors (Chen et al., 2025)	Learned stochastic	✗	✗	✓	✗
Aggregation Buffering (Lee et al., 2025)	Stabilization	N/A	N/A	N/A	N/A
SGS-GNN (Das et al., 2026)	Learned sparsif.	✓	✓	✗	N/A
MoG (Zhang et al., 2025)	Learned sparsif.	✓	✓	✓	✗
GERN (Bonchi et al., 2025)	Multi-structure	✗	✓	✓	✗
EDiS (ours)	Sparsification	✓	✓	✓	✓

Table 4: Comparison of graph sampling, sparsification, and multi-structure training methods surveyed in Section 2 and Appendix B. *Budget*: exposes an explicit target edge count or ratio. *Reusable*: the structure is computed once and reused rather than resampled from scratch every step. *Multi-struct.*: training exposes the GNN to more than one structural view. *Edge-disjoint*: multiple structures, when used, share no edges. "N/A" marks a column that does not apply to a given method.

C Formal Definitions and Construction

This appendix gives the formal definitions and exact composition rules underlying the intuitive description in Section 3.

C.1 Graph, GNN, and Edge Budget

Definition 1 (Graph)**.** Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ be a finite, loopless, undirected graph with $n = |\mathcal{V}|$ nodes, $m = |\mathcal{E}|$ edges, and $\mathbf{X} \in \mathbb{R}^{n \times d_0}$. Each undirected edge is counted once.

Definition 2 (Graph neural network)**.** A message-passing GNN updates node representations using their neighbors. A graph convolutional layer (Kipf & Welling, 2017) uses

$$\mathbf{H}_{\ell+1} = \sigma\big(\hat{\mathbf{A}}\mathbf{H}_\ell\mathbf{W}_\ell\big), \qquad \mathbf{H}_0 = \mathbf{X}, \tag{14}$$

where $\hat{\mathbf{A}}$ is normalized adjacency, $\mathbf{W}_\ell$ is a learned weight matrix, and σ is a nonlinearity.

Definition 3 (Edge budget)**.** At epoch t, nodes and features remain fixed while training uses

$$\mathcal{G}_B^{(t)} = (\mathcal{V}, \mathcal{E}_B^{(t)}, \mathbf{X}), \qquad \mathcal{E}_B^{(t)} \subseteq \mathcal{E}, \qquad |\mathcal{E}_B^{(t)}| = B. \tag{15}$$

The integer budget satisfies $0 \le B \le m$. A retention ratio $\rho \in (0, 1]$ sets $B = \lceil \rho m \rceil$. Indices k and t denote extraction rounds and training epochs.

C.2 Edge Scores and Residual Selectors

`EDiS` separates edge scoring from structural extraction.

Definition 4 (Edge score)**.** A fixed score $s : \mathcal{E} \to \mathbb{R}$ assigns larger values to preferred edges, computed once before extraction. Uniform scores set $s_e = 1$. Our default is shifted cosine similarity between endpoint features (Section 3.2).

Definition 5 (Residual selector)**.** At extraction round k, a residual selector $\mathcal{F}_k$ maps the current residual graph and edge scores to

$$\mathcal{E}_k = \mathcal{F}_k\big(\mathcal{V}, \mathcal{E}_{\text{rem}}^{(k-1)}, s\big) \subseteq \mathcal{E}_{\text{rem}}^{(k-1)}, \tag{16}$$

after which

$$\mathcal{E}_{\text{rem}}^{(k)} = \mathcal{E}_{\text{rem}}^{(k-1)} \setminus \mathcal{E}_k. \tag{17}$$

The selector may use s or ignore it and need not return a forest or a fixed number of edges.

The framework initializes the residual edge set to $\mathcal{E}_{\text{rem}}^{(0)} = \mathcal{E}$ and pairs any fixed score with any residual selector. Implemented scores include cosine, Euclidean, dot-product, uniform, and learned preferences. Implemented selectors include maximum- and minimum-weight covering forests, uniform sampling, degree-based selection, top-k neighbors, spanners, and a hybrid chaining several of these (Appendix F.1).

C.3 Maximum-Score Residual Forests

Definition 6 (Covering forest)**.** Given $H = (\mathcal{V}, E_H)$, let

$$\mathfrak{F}(H) = \{F \subseteq E_H : F \text{ is acyclic and connects every connected component of } H\}. \tag{18}$$

Definition 7 (Principal selector)**.** The default selector $\mathcal{F}_{\max}$ returns a maximum-score covering forest,

$$F^\star \in \arg\max_{F \in \mathfrak{F}(H)} \sum_{e \in F} s_e. \tag{19}$$

This forest is obtained by classical maximum-weight graphic-matroid greedy (Edmonds, 1971), equivalently by Kruskal's algorithm in decreasing score order (Kruskal, 1956). Repeating this extraction produces a partition into successive residual forests, giving the edge-disjoint decomposition and subgraph weights of Section 3.2 (Eq. (6)). It does not assert a minimum-cardinality forest partition. That stronger guarantee instead comes from matroid partition (Edmonds, 1968).

The subgraphs, their weights, and the residual pool are computed once and reused across all training epochs and all feasible budgets. Because composition (Appendix C.5) never re-invokes the selector, this holds regardless of how many budget-epoch pairs (B, t) are subsequently queried, and distinct epochs need not compose the same graph (Proposition 7, Appendix E.1).

C.4 Classical Forest Properties

Fact 1 (Bottleneck property (Kruskal, 1956; Edmonds, 1971)). *Let $e = (u, v)$ be an edge omitted by a maximum-score covering forest $F^\star$, and let $P_{F^\star}(u, v)$ denote the unique path between u and v inside the corresponding tree. Then*

$$\min_{a \in P_{F^\star}(u,v)} s_a \geq s_e. \tag{20}$$

Fact 2 (Cut certificate (Nagamochi & Ibaraki, 1992; Frank et al., 1993)). *Let $\mathcal{E}_1, \ldots, \mathcal{E}_k$ be covering forests extracted successively from the residual graph. For every nonempty proper $S \subsetneq \mathcal{V}$,*

$$|\delta_{U_k}(S)| \geq \min\left\{k, |\delta_{\mathcal{G}}(S)|\right\}. \tag{21}$$

The bottleneck property concerns a single maximum-score forest, whereas the cut certificate concerns the union of successive residual forests. Neither claims that U_k is jointly optimal among all edge sets of the same size.

C.5 Exact Stochastic Budgeted Composition

At epoch t, `EDiS` draws an ordering $\pi^{(t)} = (k_1, \ldots, k_K)$ without replacement, where, with R_r the subgraph indices remaining before draw r,

$$\Pr(k_r = k \mid R_r) = \frac{w_k}{\sum_{j \in R_r} w_j}, \qquad k \in R_r. \tag{22}$$

With $N = |U_K|$, subgraphs accumulate in sampled order until their total size first reaches or exceeds B, at round

$$r_B^{(t)} = \min\left\{r \in \{1, \ldots, K\} : \textstyle\sum_{\ell=1}^{r} |\mathcal{E}_{k_\ell}| \geq B\right\}, \tag{23}$$

giving the selected subgraphs and their candidate edges

$$A^{(t)} = \begin{cases} \varnothing, & B = 0, \\ \{k_1, \ldots, k_{r_B^{(t)}}\}, & 0 < B \leq N, \\ \{1, \ldots, K\}, & N < B \leq m, \end{cases} \qquad C_B^{(t)} = \bigsqcup_{k \in A^{(t)}} \mathcal{E}_k. \tag{24}$$

If $|C_B^{(t)}| > B$, the excess $\Delta_B^{(t)} = |C_B^{(t)}| - B$ is removed from selected subgraphs in increasing (w_k, k) order, dropping whole subgraphs until the remaining excess fits inside one subgraph, then trimming that subgraph uniformly without replacement, so at most one selected subgraph is partially retained. If $B > N$ instead, all stored edges are kept and the remaining $B - N$ edges are sampled uniformly without replacement from $\mathcal{E}_{\text{res}}$ into a residual completion set $Q_B^{(t)}$, otherwise $Q_B^{(t)} = \varnothing$.

Writing $\widetilde{\mathcal{E}}_k^{(t)}$ for the retained part of subgraph k after trimming (empty for $k \notin A^{(t)}$), the final epoch graph is

$$\boxed{\mathcal{E}_B^{(t)} = \left(\bigsqcup_{k=1}^{K} \widetilde{\mathcal{E}}_k^{(t)}\right) \sqcup Q_B^{(t)}.} \tag{25}$$

Proposition 3 (Exact-budget feasibility). *For every integer $0 \leq B \leq m$,*

$$|\mathcal{E}_B^{(t)}| = B. \tag{26}$$

The edge-disjoint partition prevents duplicates, and when $B > N$ the residual pool always contains at least $B - N$ edges because $m - N \geq B - N$. A complete proof is given in Appendix E.1.

C.6 Structural Preservation of the Stored Decomposition

To track only the edges a threshold cares about, fix $\tau \in \mathbb{R}$ and focus only on the high-score subgraph and its stored counterpart,

$$\mathcal{E}_{\geq\tau} = \{e \in \mathcal{E} : s_e \geq \tau\}, \qquad \mathcal{G}_{\geq\tau} = (\mathcal{V}, \mathcal{E}_{\geq\tau}), \qquad U_{k,\geq\tau} = U_k \cap \mathcal{E}_{\geq\tau}.$$

Theorem 1 in Section 3 states that every stored prefix certifies a lower bound on the number of threshold-qualified crossing edges within this restricted view. For every completed round, threshold, and nontrivial cut,

$$|\delta_{U_{k,\geq\tau}}(S)| \geq \min\left\{k, |\delta_{\mathcal{G}_{\geq\tau}}(S)|\right\}.$$

The exchange proof and the treatment of ties are given in Appendix E.2.

The threshold certificate above only bounds how many high-score edges survive. To state a stronger guarantee on their exact values, let $a_j(S)$ and $b_j(S)$ denote the j-th largest crossing score at cut S in $\mathcal{E}$ and in U_k, respectively.

Corollary 4 (Top-k cut-score preservation)**.** *Under the assumptions of Theorem 1,*

$$b_j(S) = a_j(S), \qquad 1 \leq j \leq \min\{k, |\delta_{\mathcal{G}}(S)|\}. \tag{27}$$

Thus every cut's top-k crossing-score values are preserved with multiplicity. The proof is given in Appendix E.3.

C.7 Retention After Budgeted Composition

The deterministic certificates above concern the stored prefix U_k. A composed epoch graph may omit or trim subgraphs, so it need not contain that prefix. Theorem 2 in Section 3 instead bounds the conditional probability that a fixed cut keeps at least one qualifying crossing edge in $\mathcal{E}_B^{(t)}$. We now state the exact identity behind that guarantee.

Condition on the sampled order $\pi^{(t)}$ together with the decomposition, subgraph weights, and budget, leaving uniform trimming or residual completion as the remaining source of randomness. We index the residual set alongside the K stored subgraphs by writing

$$D_0 = \mathcal{E}_{\mathrm{res}}, \qquad D_k = \mathcal{E}_k \ (1 \leq k \leq K), \qquad n_k = |D_k|, \qquad r_k = |D_k \cap \mathcal{E}_B^{(t)}|,$$

so $D_0, \ldots, D_K$ are the $K+1$ sets composition draws from, n_k is each one's size, and r_k counts how many of its edges the composed graph actually keeps. The retained fraction is

$$\theta_k = \begin{cases} r_k/n_k, & n_k > 0, \\ 0, & n_k = 0, \end{cases} \tag{28}$$

and, for cut S and threshold τ, the number of subgraph-k edges that would matter for that cut is

$$x_k(S,\tau) = |D_k \cap \delta_{\mathcal{G}}(S) \cap \mathcal{E}_{\geq\tau}|. \tag{29}$$

Given the sampled order, every n_k and r_k is fixed, and at most one subgraph contributes a randomly trimmed subset.

Theorem 5 (Conditional retention under composition, in full)**.** *For any fixed additive edge weight ω,*

$$\mathbb{E}\left[\omega(\mathcal{E}_B^{(t)}) \mid \pi^{(t)}\right] = \sum_{k=0}^{K} \theta_k \omega(D_k). \tag{30}$$

Moreover,

$$\Pr\left(\mathcal{E}_B^{(t)} \cap \delta_{\mathcal{G}}(S) \cap \mathcal{E}_{\geq\tau} = \varnothing \;\middle|\; \pi^{(t)}\right) = \prod_{k=0}^{K} \frac{\binom{n_k - x_k(S,\tau)}{r_k}}{\binom{n_k}{r_k}}. \tag{31}$$

Corollary 6 (Per-cut high-score edge survival)**.** *Under the same conditioning,*

$$\Pr\left(\mathcal{E}_B^{(t)} \cap \delta_{\mathcal{G}}(S) \cap \mathcal{E}_{\geq\tau} \neq \varnothing \;\middle|\; \pi^{(t)}\right) \geq 1 - \exp\left(-\sum_{k=0}^{K} \theta_k x_k(S,\tau)\right). \tag{32}$$

This is the bound displayed as Theorem 2 in Section 3.

The quantity $\sum_{k=0}^{K} \theta_k x_k(S,\tau)$ is the conditional expected number of retained high-score crossings. Neither result uses any property specific to maximum-score forests, since composition retains a uniform random subset within each partially trimmed subgraph regardless of selector. Both therefore hold unchanged for every selector in Appendix F.1. These results characterize what survives budgeted composition. They do not by themselves imply that weighted ordering is superior to uniform ordering, or that spreading edges across subgraphs is always beneficial. Those questions are tested empirically in Section 4, with proofs in Appendix E.5.

D Worked Examples of Structural Selectors

Seven selectors share the same scored input, three extraction rounds, and six-edge sample budget. Figure 8 shows the stored subgraphs and two six-edge samples from the same decomposition.

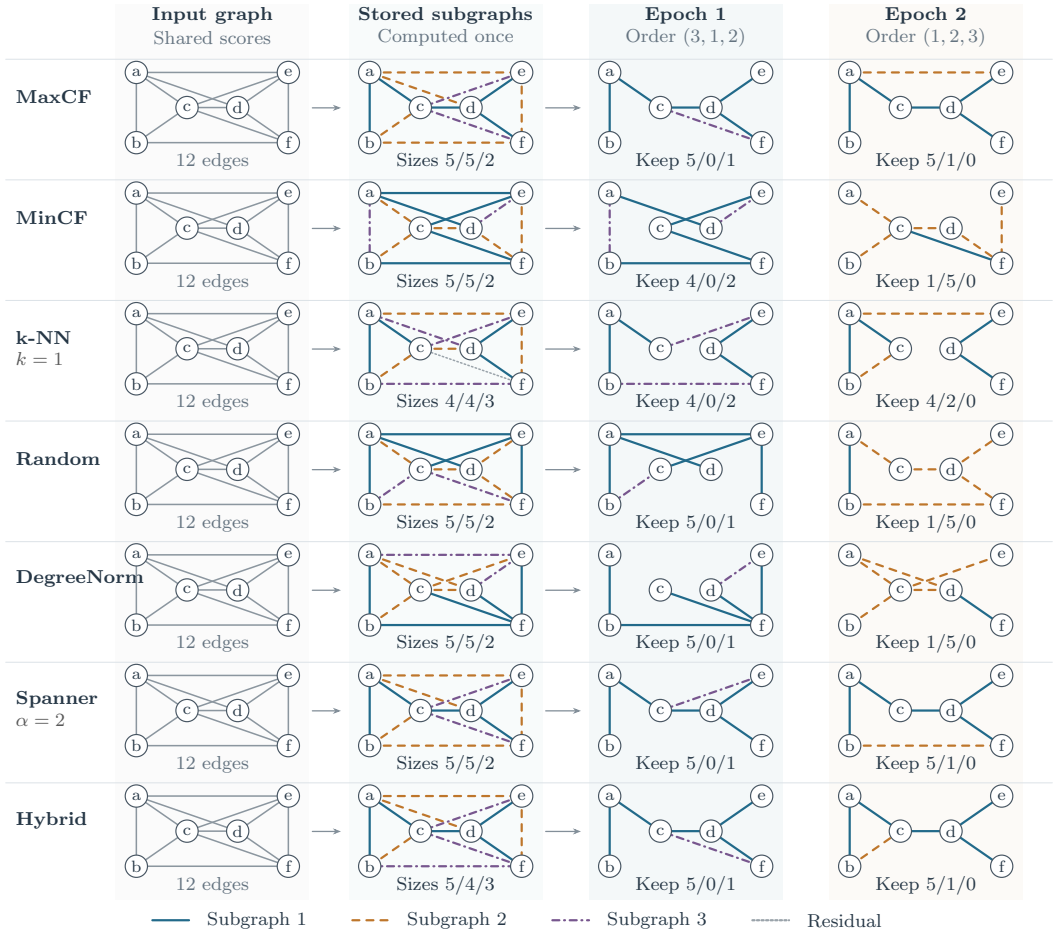


Figure 8: **Structural selection and exact-budget composition in EDiS.** Colors and line styles identify edge-disjoint subgraphs, extracted in order $(1, 2, 3)$ and reused across epochs. The two illustrative epochs sample subgraph orders $(3, 1, 2)$ and $(1, 2, 3)$, adding subgraphs until six edges are reached and trimming any excess uniformly from the lowest-weight subgraph. Samples may overlap and are not guaranteed to differ. These examples illustrate the rules, not predictive rankings.

Settings. MaxCF selects a residual covering forest with the largest total score. MinCF selects one with the smallest total score. The k-NN example uses $k = 1$ without adjusting the edge count. Random and DegreeNorm select up to five edges per round. Random gives each residual edge equal weight. DegreeNorm uses $1/d_H(u) + 1/d_H(v)$, held fixed during each round. The example spanner processes edges from shortest to longest using length $1/s_e$. It adds an edge when the current path between its endpoints is longer than $2/s_e$. This illustrative procedure differs from the randomized benchmark routine. Hybrid applies MaxCF, then 1-NN, then DegreeNorm.

The table gives the shared edge scores. Each subgraph weight is the sum of its edge scores.

Edge	ab	ac	bc	de	df	ef	cd	ae	bf	ad	ce	cf
Score	12	11	10	9	8	7	6	5	4	3	2	1

MaxCF and Spanner produce the same subgraphs in this example. The k-NN decomposition leaves one residual edge. No sample needs residual completion. Guarantees for stored subgraphs may not hold after composition. Appendix F.1 describes each selector's procedure.

E Theoretical Understanding and Proofs

A cut's crossing edges are the only channel through which one layer of message passing can move information between its two sides, so the results below characterize threshold-qualified crossings in stored maximum-score forest prefixes and their conditional retention under composition.

All graphs are finite, undirected, and loopless, with $n \geq 2$ vertices, and every undirected edge is counted once. Decomposition results apply to each realized extraction sequence satisfying the corresponding hypotheses. Ordinary cut certificates require successive residual covering forests. Score-threshold, top-k score, and score-qualified path results require maximum-score residual covering forests under fixed real scores. Randomized tie-breaking among maximizers is allowed. Arbitrary randomized or non-forest selectors do not inherit these guarantees. Sampling results condition on the realized decomposition, cached weights, and edge-budget.

E.1 Reuse and Exact-Budget Feasibility

Proposition 7 (Cacheable stochastic support generation)**.** *Fix a stored decomposition $\mathcal{D}$. Any number of subsequent edge-budget–epoch queries can be answered without reevaluating the structural selector. For fixed B, let μ_B be the support distribution induced by composition. If epochs are independent draws from this same distribution and μ_B is non-degenerate, then for $t \neq t'$,*

$$\Pr(\mathcal{E}_B^{(t)} \neq \mathcal{E}_B^{(t')}) = 1 - \sum_{\mathcal{S}} \mu_B(\mathcal{S})^2 > 0. \tag{33}$$

Proof. The selector is invoked only during decomposition, while composition operates on the stored partition and weights. Independent draws from μ_B coincide with probability $\sum_{\mathcal{S}} \mu_B(\mathcal{S})^2$. Taking its complement proves the identity. Non-degeneracy makes this collision probability strictly less than one. □

Proof of Proposition 3. The stored subgraphs and residual form an edge-disjoint partition of $\mathcal{E}$, so every returned edge is original and no edge is counted twice. For $B = 0$, composition returns the empty set. If $0 < B \leq |U_K|$, the selected subgraphs reach the edge-budget and trimming removes exactly $|C_B^{(t)}| - B$ edges. If $|U_K| < B \leq m$, all structural subgraphs are retained and the residual supplies the deficit, since $|\mathcal{E}_{\mathrm{res}}| = m - |U_K| \geq B - |U_K|$. These cases include $K = 0$ and yield exactly B edges. □

E.2 Score-Threshold Certificate

Throughout this proof, F_i denotes the i-th extracted subgraph $\mathcal{E}_i$ and U_k its stored union $\bigcup_{i \leq k} \mathcal{E}_i$, matching the maximum-score covering-forest notation of Fact 2 under the default $\mathcal{F}_{\max}$ selector.

Proof of Theorem 1. Fix τ and let $F_i^\tau := F_i \cap \mathcal{E}_{\geq \tau}$.

Suppose F_i^τ does not span a component of $\big(\mathcal{V}, \mathcal{E}_{\mathrm{rem}}^{(i-1)} \cap \mathcal{E}_{\geq \tau}\big)$. Then some $e \in \mathcal{E}_{\mathrm{rem}}^{(i-1)} \cap \mathcal{E}_{\geq \tau}$ joins two components of F_i^τ. Since F_i is a covering forest of the residual graph, $F_i + e$ contains a unique cycle, and since e creates no cycle in F_i^τ, that cycle contains some $f \in F_i \setminus F_i^\tau$, so $s_f < \tau \leq s_e$. Hence

$$s(F_i - f + e) = s(F_i) - s_f + s_e > s(F_i), \tag{34}$$

contradicting the maximum-score optimality of F_i. Therefore

$$F_i^\tau \in \mathfrak{F}\big(\mathcal{V},\ \mathcal{E}_{\mathrm{rem}}^{(i-1)} \cap \mathcal{E}_{\geq \tau}\big). \tag{35}$$

Also

$$\mathcal{E}_{\mathrm{rem}}^{(i-1)} \cap \mathcal{E}_{\geq \tau} = \mathcal{E}_{\geq \tau} \setminus \bigcup_{j<i} F_j^\tau, \tag{36}$$

so $F_1^\tau, \ldots, F_k^\tau$ are successive residual covering forests of $\mathcal{G}_{\geq \tau}$ for every completed round $1 \leq k \leq K$. Applying Fact 2 inside $\mathcal{G}_{\geq \tau}$ to $U_{k,\geq \tau} = \bigcup_{i \leq k} F_i^\tau$,

$$\boxed{|\delta_{U_{k,\geq \tau}}(S)| \ \geq\ \min\{k, |\delta_{\mathcal{G}_{\geq \tau}}(S)|\}.} \tag{37}$$

□

This exchange uses maximum-score optimality alone, so no sign assumption on s and no tie-breaking rule enter the argument. A non-maximum-score covering forest still gives the ordinary count certificate, but not the score-threshold form above.

The guarantee holds simultaneously for every nontrivial cut in the stored prefix. Its crossing edges provide potential one-hop message-passing connections between the two sides. Whether the retained connections carry useful evidence for node classification depends on the features, task, and model. The certificate itself concerns edge scores and cut counts.

E.3 Top-k Cut-Score Corollary

For an edge set A, define the sum of its largest crossing scores by

$$T_k(S;A) = \sum_{j=1}^{\min\{k,|\delta_A(S)|\}} s_{(j)}(\delta_A(S)).$$

For a graph, A denotes its edge set. Scores are ordered with multiplicity, so tied edges contribute separately. Let $a_1(S) \geq a_2(S) \geq \cdots$ be the scores of the original crossing edges of a cut S and $b_1(S) \geq b_2(S) \geq \cdots$ those kept in U_k.

Corollary 8 (Top-k cut-score preservation, restated)**.** *For every completed round $1 \leq k \leq K$ and nonempty proper cut S,*

$$b_j(S) = a_j(S), \qquad 1 \leq j \leq \min\{k, |\delta_{\mathcal{G}}(S)|\}. \tag{38}$$

Proof of Corollary 8. Fix S and write $M = |\delta_{\mathcal{G}}(S)|$. For $j \leq \min\{k, M\}$ set $\tau = a_j(S)$. Then

$$|\delta_{\mathcal{G}_{\geq\tau}}(S)| \geq j, \tag{39}$$

so by Theorem 1, $|\delta_{U_{k,\geq\tau}}(S)| \geq j$, giving $b_j(S) \geq a_j(S)$. Since $U_k \subseteq \mathcal{E}$, also $b_j(S) \leq a_j(S)$. Hence

$$\boxed{b_j(S) = a_j(S), \qquad j \leq \min\{k, M\}.} \tag{40}$$

Taking τ below every score gives $M \geq |\delta_{U_k}(S)| \geq \min\{k, M\}$, so $\min\{k, |\delta_{U_k}(S)|\} = \min\{k, M\}$, and summing (40) term by term,

$$\boxed{T_k(S;U_k) = T_k(S;\mathcal{G}).} \tag{41}$$

□

The equality preserves the top-k crossing-score values with multiplicity. With tied scores, it need not preserve particular edge identities. For the cosine default, this means preserving the largest feature-similarity scores across each cut. The usefulness of that preference for prediction is evaluated empirically.

E.4 Equivalent Interpretations

Proposition 9 (Submodular reading (Nemhauser et al., 1978))**.** *For a fixed cut S and integer $k \geq 1$, let $g(A) = |\delta_A(S)|$ and $\phi(z) = \min\{z, k\}$. Then $f_{S,k} := \phi \circ g$ is monotone submodular. If $1 \leq k \leq K$ and U_k is a prefix of successive residual covering forests, then $f_{S,k}(U_k) = f_{S,k}(\mathcal{E})$.*

Proof. g is modular and ϕ is nondecreasing and concave, so

$$f_{S,k}(A) = \min\{|\delta_A(S)|, k\} \tag{42}$$

is monotone submodular, and so is any nonnegative combination $\sum_S \alpha_S f_{S,k}$. Fact 2 gives

$$\boxed{f_{S,k}(U_k) = f_{S,k}(\mathcal{E}).} \tag{43}$$

□

This is an interpretation of the stored certificate, not the objective `EDiS` optimizes.

Proposition 10 (Edge-disjoint path reading)**.** *Under a pure $\mathcal{F}_{\max}$ schedule with fixed real scores, let $1 \leq k \leq K$ and $e = (u, v) \in \mathcal{E} \setminus U_k$. Then U_k contains k edge-disjoint u–v paths whose edges all have score at least s_e.*

Proof. $e \notin U_k$ gives

$$e \in \mathcal{E}_{\text{rem}}^{(i-1)} \qquad \forall\, i \leq k. \tag{44}$$

By the bottleneck property (Fact 1), each F_i carries a path $P_i(u, v) \subseteq F_i$ with $\min_{f \in P_i(u,v)} s_f \geq s_e$. Since $F_i \cap F_j = \varnothing$ for $i \neq j$, the paths $P_1, \ldots, P_k$ are edge-disjoint. Hence U_k holds k edge-disjoint u–v paths of score at least s_e. □

In plain terms, for any original edge e outside the stored prefix U_k, the stored prefix U_k already contains k different routes between its two endpoints, no two of which share an edge, and every edge on every route scores at least as well as e itself. Dropping e cannot disconnect u from v through routes this good unless at least k of those routes are also removed. This is the path-side mirror of the cut certificate above. The cut version counts surviving edges across a fixed cut, while the path version exhibits the actual routes behind that count. Two of the k paths may still share an intermediate vertex, since the guarantee covers edges rather than vertices, and it holds only for the pure $\mathcal{F}_{\max}$ schedule with fixed real scores.

These paths establish alternative routes in the stored prefix. The proposition does not bound their hop lengths or their survival after composition. A path can support propagation in an L-layer message-passing GNN only

if its length is at most L and all its edges are retained. Even then, reachability alone does not guarantee useful influence on the prediction.

E.5 Composition Retention

All probabilities condition on the realized decomposition, cached subgraph weights, and edge-budget in addition to the displayed $\pi^{(t)}$. Trimming uses increasing (w_k, k) order, so this conditioning fixes all retained counts. For $B = 0$, the identities hold trivially. The displayed ordering may then be regarded as an unused latent weighted permutation. For $K = 0$, this permutation is empty. Write $A^{(t)}$ for the selected structural indices. For $n_k = 0$, set $r_k = 0$ and $\theta_k = 0$. We use $\binom{0}{0} = 1$ and $\binom{a}{b} = 0$ for integers $b > a \geq 0$. Thus an empty pool contributes a factor of one to cut avoidance.

Fix $\mathcal{D}$, its weights, and B, and condition on $\pi^{(t)}$. Set $D_0 = \mathcal{E}_{\text{res}}$ and $D_k = \mathcal{E}_k$ for $1 \leq k \leq K$, with $n_k = |D_k|$, retained count r_k, and $\theta_k = r_k/n_k$ for $n_k > 0$ (zero otherwise). For a cut S and threshold τ, let $x_k(S, \tau) = |D_k \cap \delta_{\mathcal{G}}(S) \cap \mathcal{E}_{\geq\tau}|$ count its high-score crossing edges.

Theorem 11 (Conditional retention under composition, restated)**.** *For any fixed additive edge weight ω, cut S, and threshold τ,*

$$\mathbb{E}[\omega(\mathcal{E}_B^{(t)}) \mid \pi^{(t)}] = \sum_{k=0}^{K} \theta_k\, \omega(D_k), \qquad \Pr\big(\mathcal{E}_B^{(t)} \cap \delta_{\mathcal{G}}(S) \cap \mathcal{E}_{\geq\tau} = \varnothing \mid \pi^{(t)}\big) = \prod_{k=0}^{K} \frac{\binom{n_k - x_k(S,\tau)}{r_k}}{\binom{n_k}{r_k}}. \tag{45}$$

Proof of Theorem 11. Condition on $\pi^{(t)}$, fixing $A^{(t)}$ and the retained counts $0 \leq r_k \leq n_k$, $0 \leq k \leq K$, with $\sum_{k=0}^{K} r_k = B$. Depletion removes whole subgraphs before partially trimming at most one subgraph. Uniform deletion leaves a uniform retained subset of that subgraph. All other structural pool subsets are deterministic. Residual completion occurs only when every structural subgraph is fully retained, so only the residual subset can then be non-degenerate. Thus at most one pool subset is non-degenerate under the conditioning, and the pool subsets are conditionally independent. Each pool contributes a uniform r_k-subset, including the unique empty or full subset when $r_k = 0$ or $r_k = n_k$. Hence

$$\Pr(e \in \mathcal{E}_B^{(t)} \mid \pi^{(t)}) = \theta_k \qquad \forall\, e \in D_k. \tag{46}$$

By linearity of expectation on $\omega(\mathcal{E}_B^{(t)}) = \sum_e \omega_e \mathbf{1}\{e \in \mathcal{E}_B^{(t)}\}$,

$$\boxed{\mathbb{E}\big[\omega(\mathcal{E}_B^{(t)}) \mid \pi^{(t)}\big] = \sum_{k=0}^{K} \theta_k\, \omega(D_k).} \tag{47}$$

For a cut S and threshold τ, mark the $x_k(S, \tau) = |D_k \cap \delta_{\mathcal{G}}(S) \cap \mathcal{E}_{\geq\tau}|$ crossing edges in each D_k. Exactly $\binom{n_k - x_k(S,\tau)}{r_k}$ of the $\binom{n_k}{r_k}$ retained subsets avoid them, so

$$h_k := h\big(n_k, r_k, x_k(S,\tau)\big) = \frac{\binom{n_k - x_k(S,\tau)}{r_k}}{\binom{n_k}{r_k}} \tag{48}$$

is the probability that subgraph k retains none of them. Independence across subgraphs gives

$$\boxed{\Pr\big(\text{no retained } \tau\text{-qualified crossing of } S \mid \pi^{(t)}\big) = \prod_{k=0}^{K} h_k.} \tag{49}$$

□

A vertex is isolated exactly when its singleton cut is empty, so $S = \{v\}$ with τ below every score gives its isolation probability. An isolated vertex receives no neighbor messages that epoch, so this same identity also bounds how often a node's aggregation degenerates to its own features alone. These identities bound retention at each cut individually rather than the joint event of every cut retaining an edge, so connectivity of $\mathcal{G}_B^{(t)}$ at a given B is a separate, cut-by-cut question.

Corollary 12 (High-score cut survival, restated)**.** *Under the same conditions,*

$$\Pr\big(\mathcal{E}_B^{(t)} \cap \delta_{\mathcal{G}}(S) \cap \mathcal{E}_{\geq\tau} \neq \varnothing \mid \pi^{(t)}\big) \geq 1 - \exp\Big(-\textstyle\sum_{k=0}^{K} \theta_k x_k(S,\tau)\Big). \tag{50}$$

Proof of Corollary 12. For $n = 0$ or $r = 0$ the factor and its exponential bound are one. For $n > 0$ and $r > n - x$, $h(n, r, x) = 0$ and the bound is immediate. Otherwise $0 \leq r \leq n - x$ and

$$h(n, r, x) = \frac{\binom{n-x}{r}}{\binom{n}{r}} = \prod_{i=0}^{r-1} \frac{n - x - i}{n - i}. \tag{51}$$

Since $\frac{n-x-i}{n-i} \le 1 - \frac{x}{n}$ for $0 \le i < n$,

$$h(n, r, x) \le \Big(1 - \frac{x}{n}\Big)^r \le e^{-rx/n}. \tag{52}$$

Hence

$$\prod_{k=0}^{K} h_k \le \exp\Big(-\sum_{k=0}^{K} \theta_k x_k(S, \tau)\Big), \tag{53}$$

and complementing (49),

$$\boxed{\Pr\big(\exists \text{ retained edge crossing } S \text{ with } s_e \ge \tau \mid \pi^{(t)}\big) \;\ge\; 1 - \exp\Big(-\sum_{k=0}^{K} \theta_k x_k(S, \tau)\Big).} \tag{54}$$

□

Writing $\mu = \sum_{k=0}^{K} \theta_k x_k(S, \tau)$, the bound gives conditional survival probability at least $1 - \delta$ when $\mu \ge \log(1/\delta)$, for $0 < \delta < 1$. It is weak for small μ and equals zero when $\mu = 0$. This is a fixed-cut statement, not a simultaneous connectivity or path-survival guarantee.

E.6 Optional Composition Variants

Two theory-driven variants extend the composition step. Both are analyzed here as additional theoretical guarantees available within the framework, alongside rather than in place of the main experiments.

Certificate-core composition. Under the maximum-score residual-forest assumptions, set $U_0 = \varnothing$. For edge-budget B let $q(B) = \max\{q \in \{0, \dots, K\} : |U_q| \le B\}$ and always retain the prefix $U_{q(B)}$, filling the remaining $B - |U_{q(B)}|$ edges stochastically without replacement from $\mathcal{E} \setminus U_{q(B)}$. Since $\mathcal{E}_B^{(t)} \supseteq U_{q(B)}$ and $\delta_A(S)$ is monotone in A, Fact 2 and Theorem 1 give every epoch graph the deterministic guarantees

$$\boxed{|\delta_{\mathcal{E}_B^{(t)}}(S)| \ge \min\{q(B), |\delta_{\mathcal{G}}(S)|\}, \qquad |\delta_{\mathcal{E}_B^{(t)}}(S) \cap \mathcal{E}_{\ge\tau}| \ge \min\{q(B), |\delta_{\mathcal{G}_{\ge\tau}}(S)|\}.} \tag{55}$$

This trades cross-epoch variability for a deterministic per-epoch certificate.

Proposition 13 (Average-score trimming is expectation-optimal)**.** *Fix available nonempty subgraphs indexed by k, let $\bar{s}_k = s(\mathcal{E}_k)/n_k$, and let $0 \le B \le \sum_k n_k$ be an integer. Among allocations $r_k \in \{0, \dots, n_k\}$ with $\sum_k r_k = B$, uniform within-subgraph retention maximizes $\mathbb{E}\big[\sum_k s(\widetilde{\mathcal{E}}_k)\big] = \sum_k r_k \bar{s}_k$ by filling subgraphs in decreasing order of $\bar{s}_k$, with at most one subgraph partially filled. Other allocations can tie when subgraphs have equal average score. The main method instead trims by cached total weight w_k rather than this expectation-optimal average-score rule. That w_k-based greedy fill is equally deterministic in its own retained counts given fixed tie-breaking, though uniform sampling within the one partially filled subgraph can still vary which edges are kept. This score-optimality statement does not establish an accuracy-optimal allocation. The main method's weighted ordering and total-weight trimming are evaluated through the empirical ablations.*

Proof. Uniform retention gives $\mathbb{E}[s(\widetilde{\mathcal{E}}_k)] = r_k \bar{s}_k$, so $\sum_k r_k \bar{s}_k$ is linear in $(r_k)_k$ subject to $0 \le r_k \le n_k$ and $\sum_k r_k = B$. If an optimum has $r_a < n_a$, $r_b > 0$, and $\bar{s}_a > \bar{s}_b$, moving one unit from b to a changes the objective by $\bar{s}_a - \bar{s}_b > 0$, contradicting optimality. Hence

$$\boxed{\text{an optimum fills subgraphs in decreasing } \bar{s}_k \text{ order, at most one subgraph split.}} \tag{56}$$

□

The results establish exact-budget reuse for any cached partition and deterministic cut and score-qualified path certificates for maximum-score forest prefixes. They also characterize conditional edge retention and fixed-cut survival under stochastic composition. The last property does not transfer the full prefix certificates to every epoch graph. The optional certificate-core variant retains a prefix chosen according to B and therefore inherits its deterministic cut guarantees.

These combinatorial results describe available graph connections without assuming a particular GNN or loss. The benchmark ablations provide complementary evidence for predictive utility. At $\rho = 0.3$, the reported means favor `EDiS`-Lite over Frozen on 13 of 14 datasets, Direct on 11, and Shuffled groups on 12 (Table 19). This supports the practical contribution of decomposition and resampling in the tested settings, while leaving open how much of the gain is specifically attributable to the cut and path certificates rather than other changes in the induced sampling distribution.

F Selectors and Computational Cost

F.1 Family of Residual Selectors

Each of the seven selectors below implements the same residual-selector interface $\mathcal{F}$ from a residual graph H and its fixed edge scores s, so any one of them can be dropped into `EDiS` without changing anything else in the pipeline. What differs is only how each one decides which edges to keep, and consequently its round cost $T_i(n, m_i)$, explained selector by selector below and summarized in Table 5 at the end of this appendix.

Maximum/minimum covering forest (`maxst`/`minst`, MaxCF/MinCF in Section 4). The implementation names `maxst`/`minst` reflect that each round's forest spans every component of the current residual graph. We call it a *covering forest* instead because that spanning property holds only within one round, not for the budget-limited graph actually composed for training (Section 3.4). This selector puts every remaining edge in order by score, either strongest first or weakest first, and adds edges one at a time down that list, skipping any edge that would close a loop back to nodes already connected. This is Kruskal's algorithm (Kruskal, 1956; Edmonds, 1971) run on our scores instead of raw edge weights, with ties kept in their original order. Because it only adds loop-free edges, it ends up touching every part of the residual graph. `maxst` and `minst` differ only in which direction we sort. Each round sorts the remaining edges and tracks which nodes are already connected, costing a little more than one plain scan.

k-nearest neighbor (`knn`). This selector lets every node claim its k best-scoring remaining edges, or all of them if fewer than k are left, and keeps the union of every node's claims. Asking for a fixed total M instead of a fixed k trims or tops up that union by score until it hits M exactly, which can take away an edge a node had already claimed, so no node is guaranteed to keep its own top choices. This touches each remaining edge from both ends.

Graph spanner (`spanner`). This selector works on each disconnected piece of the residual graph on its own, building a smaller subgraph using edge lengths that shrink as the score grows, so the best-scoring edges look shortest. The result must keep every pair of nodes within a stretch factor α of their true distance in these lengths (Peleg & Schäffer, 1989), a bound on path length rather than hop count. Construction is randomized, and its cost depends on the spanner routine used.

Degree-normalized (`degree_norm`). This selector draws a set number M of edges at random without repeats, weighting each edge uv by $w_{uv} = 1/d_H(u) + 1/d_H(v)$ so edges touching low-degree nodes are more likely to be picked, recomputing degrees every round. It ignores the score entirely and guarantees neither connectivity nor a minimum degree. One pass over the edges is enough to compute the weights.

Random (`random`). This selector shuffles the remaining edges uniformly at random and keeps the first M. Ignoring both score and structure, it serves as the score-free baseline the others are compared against, and needs only one pass over the edges.

Scheduled (Hybrid). A schedule chains several selectors together, using a different one each round on whatever edges remain, and repeating the last one once the list runs out. Any of the five selectors above can appear, in any order, and can repeat, so this is a general way to combine selectors rather than one fixed recipe. The schedule used in our experiments runs `maxst`, then `knn`, then `degree_norm` for every later round, first extracting a connectivity-preserving backbone, then giving each node its own best local edges, then spreading coverage to whichever nodes are still thin. It stops under the same conditions as the other selectors.

Selectors outside this family. Effective-resistance sampling (Spielman & Srivastava, 2008) needs spectral information recomputed as the residual changes. Forest-fire sampling (Leskovec & Faloutsos, 2006), SCAN (Xu et al., 2007), and rank-degree sampling (Voudigari et al., 2016) likewise need traversal, clustering, or ranking steps beyond a residual degree pass. None of these are implemented here as candidate selectors inside `EDiS`'s own construction, though several are compared against `EDiS` directly as full competing methods in Section 4.

F.2 Cost of Construction and Epoch Composition

We now analyze where EDiS's computational cost is spent, writing $n = |\mathcal{V}|$, $m = |\mathcal{E}|$, and d_0 for the input feature dimension. Every edge is scored once, before construction begins, at a cost of $T_{\text{score}} = O(md_0)$, and this cost cannot be skipped even by a score-free selector, since the same scores also determine subgraph importance.

Building the decomposition runs the selector once per round over whatever edges remain, so its total cost is just the sum of each round's selector cost and residual bookkeeping. For the repeated-sorting maxst construction, writing m_i for the residual edge count at round i, this gives

$$T_{\text{construct}} = O\left(Kn + \sum_{i=1}^{K} m_i \log(m_i + 1)\right) \subseteq O(Kn + Km \log(m + 1)) .$$

The dependence on K can be absorbed only when the round cap is treated as a fixed constant. Fixed scores also permit implementations that reuse an initial ordering. Storing the resulting subgraphs and the residual adds only $O(m + K)$ extra space, on top of the graph itself and its features.

Composition reuses the partition and scores without repeating extraction. An implementation can determine retained counts from cached group sizes before materializing edges. With an $O(K \log(K + 1))$ weighted ordering and trimming-order sort, random-access pools, and uniform subset sampling in time proportional to the returned subset size, this yields $O(K \log(K + 1) + B)$ composition time. This is an implementation-dependent bound. Copying entire candidate pools or generating full-pool permutations can introduce additional cost depending on their sizes, up to m. Expanding the chosen edges into their reciprocal directed pairs for message passing adds a further $T_{\text{reciprocal}} = O(B)$ per epoch.

Summing over all T training epochs, the total cost of a job is the one-time scoring and construction cost plus the per-epoch composition, reciprocal-expansion, training, transfer, and validation costs added up across epochs,

$$T_{\text{job}} = T_{\text{score}} + T_{\text{construct}} + \sum_{t=1}^{T} \left(T^{(t)}_{\text{compose}} + T^{(t)}_{\text{reciprocal}} + T^{(t)}_{\text{train}} + T^{(t)}_{\text{transfer}} + T^{(t)}_{\text{val}} \right).$$

Full-graph validation and test inference, where used, process the original graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ instead of a composed one, so their cost depends on (n, m) rather than B, and is charged separately from training. The headline point is that scoring and construction are paid only once, while composition and reciprocal expansion recur every epoch. Budget-proportional edge materialization is possible under the implementation conditions above. Measured runtime is reported separately in Appendix H.

Selector	**Round cost** $T_i(n, m_i)$	**Structural guarantee**
maxst / minst	$O(n + m_i \log(m_i + 1))$	covering forest of every component
knn	$O(m_i \log(m_i + 1))$	top-k per node, unless trimmed to hit M
spanner	depends on the spanner routine used	distance stretch at most α
degree_norm	$O(m_i)$	none
random	$O(m_i)$	none
Hybrid	cost of whichever selector is active that round	inherits from the active selector

Table 5: Round cost and structural guarantee of each residual selector from Section F.1.

G Experimental Details

G.1 Dataset

The experiments use 19 node-classification benchmarks, spanning six heterophilic small/medium graphs, eight homophilic small/medium graphs, and five large-scale graphs (Table 6). Node and edge counts, and average degree, are computed after treating each graph as undirected and removing self-loops. Adjusted homophily measures how often connected nodes share a class after correcting for class proportions (Platonov et al., 2023). Classes/Tasks reports the number of prediction classes for every dataset except ogbn-proteins, a multi-label dataset with 112 binary prediction tasks, and Features gives the input feature dimension. Most datasets are evaluated with accuracy. Minesweeper, Questions, and ogbn-proteins use AUC instead, since their prediction tasks are binary.

Dataset	Graph size			Structure	Prediction task		
	Nodes	Edges	Avg. degree	Adj. homophily	Classes/Tasks	Features	Metric
Heterophilic small/medium (Platonov et al., 2023)							
Squirrel	2.22K	47.00K	42.28	0.01	5	2.09K	Acc
Chameleon	890	8.85K	19.90	0.03	5	2.33K	Acc
Roman-Empire	22.66K	32.93K	2.91	-0.05	18	300	Acc
Amazon-Ratings	24.49K	93.05K	7.60	0.14	5	300	Acc
Minesweeper	10.00K	39.40K	7.88	0.01	2	7	AUC
Questions	48.92K	153.54K	6.28	0.02	2	301	AUC
Homophilic small/medium							
Cora (Yang et al., 2016)	2.71K	5.28K	3.90	0.77	7	1.43K	Acc
CiteSeer (Yang et al., 2016)	3.33K	4.55K	2.74	0.67	6	3.70K	Acc
PubMed (Yang et al., 2016)	19.72K	44.32K	4.50	0.69	3	500	Acc
Amazon-Computer (Shchur et al., 2018)	13.75K	245.86K	35.76	0.68	10	767	Acc
Amazon-Photo (Shchur et al., 2018)	7.65K	119.08K	31.13	0.79	8	745	Acc
Coauthor-CS (Shchur et al., 2018)	18.33K	81.89K	8.93	0.78	15	6.81K	Acc
Coauthor-Physics (Shchur et al., 2018)	34.49K	247.96K	14.38	0.87	5	8.42K	Acc
WikiCS (Mernyei & Cangea, 2020)	11.70K	215.60K	36.85	0.58	10	300	Acc
Large-scale							
ogbn-arxiv (Hu et al., 2020)	169.34K	1.17M	13.77	0.42	40	128	Acc
ogbn-proteins (Hu et al., 2020)	132.53K	39.56M	597.00	0.05	112	8	AUC
ogbn-products (Hu et al., 2020)	2.45M	61.86M	50.52	0.46	47	100	Acc
Pokec (Lim et al., 2021)	1.63M	22.30M	27.32	0.42	2	65	Acc
Reddit (Hamilton et al., 2017)	232.97K	57.31M	492.00	0.74	41	602	Acc

Table 6: Dataset statistics for the 19 node-classification benchmarks used in the current experiments.

G.2 Baseline Settings

We compare EDiS against baselines drawn from three families, grouped by the structural signal each relies on rather than by implementation detail. Anchors bound the comparison without adapting to the graph, and comprise a feature-only MLP (No Graph), a Spanning Tree, and Full Graph with the same tuned GNN configuration (Luo et al., 2024). Semantic baselines learn which edges to keep jointly with the GNN parameters, and comprise Unified-LTH (Chen et al., 2021), AdaGLT (Zhang et al., 2024), and MoG (Zhang et al., 2025). Topological baselines score edges directly from the graph, independent of the downstream task, and comprise Random, Local Degree (Hamann et al., 2016), Rank Degree (Voudigari et al., 2016), Forest Fire (Leskovec & Faloutsos, 2006), SCAN (Xu et al., 2007), L-Spar and G-Spar (Satuluri et al., 2011), L-Sim (Chen et al., 2023), Spectral (ER) (Spielman & Srivastava, 2008), t-Spanner (Peleg & Schäffer, 1989), and DSpar (Liu et al., 2023).

Sampling-based related methods. Neighborhood- and subgraph-sampling methods such as GraphSAGE (Hamilton et al., 2017), GraphSAINT (Zeng et al., 2020), and Cluster-GCN (Chiang et al., 2019) control a different resource, the minibatch or neighborhood size used for scalability, rather than a global edge-retention budget. They are therefore not part of the matched-edge-budget comparison, and GraphSAGE appears only as the alternate backbone in RQ7.

G.3 Propagation and Evaluation Settings

For full-batch GCN training, we add each retained undirected edge in both directions. We add one self-loop per node. The edge-budget B counts only the undirected non-loop edges. All layers use the same epoch graph, with normalized adjacency $\hat{\mathbf{A}}^{(t)} = (\tilde{\mathbf{D}}^{(t)})^{-1/2}(\mathbf{A}^{(t)} + \mathbf{I})(\tilde{\mathbf{D}}^{(t)})^{-1/2}$. Here $\tilde{\mathbf{D}}^{(t)}$ holds the degrees of $\mathbf{A}^{(t)} + \mathbf{I}$. We recompute normalization for the current topology. Edge scores guide selection. They are not propagation weights. The architecture and training objective stay the same.

G.4 Hyperparameter Settings

Every baseline and sparsifier method uses the same per-dataset training setup for a given backbone. This keeps accuracy differences tied to the sparsification method, not the training setup. Per-dataset settings follow the defaults of Luo et al. (2024) for every dataset except Reddit, which that work does not include. We use two backbones. GCN is the default for every method. GraphSAGE is used only by Full GraphSAGE and EDiS-GraphSAGE. In both tables below, L is the number of GNN layers, H is the hidden dimension, LR is the learning rate, and WD is the weight decay.

$K_{\max}$. The round cap $K_{\max}$ on subgraph extraction is itself a hyperparameter, chosen per dataset rather than fixed globally, and applied the same way regardless of backbone. Every small- and medium-scale dataset, homophilic or heterophilic, uses $K_{\max} = 10$. The five large-scale graphs instead use $K_{\max} = 80$ for Pokec, $K_{\max} = 40$ for ogbn-arxiv, $K_{\max} = 20$ for Reddit, and $K_{\max} = 10$ for ogbn-products and ogbn-proteins. These are the same values used throughout the main experiments and the extended ablations, where Section 4 (RQ6) and Appendix I.7 study how sensitive results are to this choice.

MLP edge score. The learned score used in the edge-score ablation (Section 4 RQ5, Appendix I.2) trains a small MLP on the concatenated features of each edge's two endpoints to predict an edge score. Training uses only edges whose two endpoints both carry a training label, with target 1 if the endpoints share a label and 0 otherwise, and an equal number of same-label and different-label edges sampled per batch. PubMed's semi-supervised split labels too few training nodes to assemble this balanced sample, so the MLP score is marked N/A for PubMed throughout.

	Protocol		Architecture			Optimization			Evaluation
Dataset	**Epochs**	**Runs**	L	H	**Norm./Res.**	**Dropout**	**LR**	**WD**	**Metric**
Heterophilic small/medium									
Squirrel	500	10	4	256	bn, res	0.70	1e-2	5e-4	Acc
Chameleon	200	10	5	512	–	0.20	5e-3	1e-3	Acc
Roman-Empire	2500	3	9	512	bn, res	0.50	1e-3	0	Acc
Amazon-Ratings	2500	3	4	512	bn, res	0.50	1e-3	0	Acc
Minesweeper	2000	3	12	64	bn, res	0.20	1e-2	0	AUC
Questions	1500	3	10	512	res	0.30	3e-5	0	AUC
Homophilic small/medium									
Cora	500	5	3	512	–	0.70	1e-3	5e-4	Acc
CiteSeer	500	5	2	512	–	0.50	1e-3	1e-2	Acc
PubMed	500	5	2	256	–	0.70	5e-3	5e-4	Acc
Amazon-Computer	1000	3	3	512	ln	0.50	1e-3	5e-5	Acc
Amazon-Photo	1000	3	6	256	ln, res	0.50	1e-3	5e-5	Acc
Coauthor-CS	1500	3	2	512	ln, res	0.30	1e-3	5e-4	Acc
Coauthor-Physics	1500	3	2	64	ln, res	0.30	1e-3	5e-4	Acc
WikiCS	1000	3	3	256	ln	0.50	1e-3	0	Acc
Large-scale									
ogbn-arxiv	2000	2	5	512	bn, res	0.50	5e-4	5e-4	Acc
ogbn-proteins	100	2	3	80	–	0.25	1e-2	0	AUC
ogbn-products	300	2	5	200	ln	0.50	3e-3	0	Acc
Pokec	2000	2	7	256	bn, res	0.20	5e-4	0	Acc
Reddit	300	2	5	200	ln	0.50	3e-3	0	Acc

Table 7: Per-dataset training configuration for the GCN backbone.

	Protocol		Architecture			Optimization			Evaluation
Dataset	**Epochs**	**Runs**	L	H	**Norm./Res.**	**Dropout**	**LR**	**WD**	**Metric**
Heterophilic small/medium									
Squirrel	500	10	3	256	bn, res	0.70	1e-2	–	Acc
Chameleon	200	10	4	256	bn, res	0.70	1e-2	–	Acc
Roman-Empire	2500	3	9	256	bn	0.30	1e-3	–	Acc
Amazon-Ratings	2500	3	9	512	bn, res	0.50	1e-3	–	Acc
Minesweeper	2000	3	15	64	bn, res	0.20	1e-2	–	AUC
Questions	1500	3	6	512	ln	0.20	1e-3	–	AUC
Homophilic small/medium									
Cora	500	5	3	256	–	0.70	1e-3	–	Acc
CiteSeer	500	5	3	512	–	0.20	1e-3	–	Acc
PubMed	500	5	4	512	–	0.70	5e-3	–	Acc
Amazon-Computer	1000	3	4	64	ln	0.30	1e-3	–	Acc
Amazon-Photo	1000	3	6	64	ln, res	0.20	1e-3	–	Acc
Coauthor-CS	1500	3	2	512	ln, res	0.50	1e-3	–	Acc
Coauthor-Physics	1500	3	2	64	bn, res	0.70	1e-3	–	Acc
WikiCS	1000	3	2	256	ln	0.70	1e-3	–	Acc
Large-scale									
ogbn-arxiv	2000	2	4	256	bn, res	0.50	5e-4	–	Acc
ogbn-proteins	1000	2	6	512	bn, res	0.30	1e-2	–	AUC
ogbn-products	1000	2	5	256	ln	0.50	3e-3	–	Acc
Pokec	2000	2	7	256	bn, res	0.20	5e-4	–	Acc
Reddit	300	2	5	200	ln	0.50	3e-3	–	Acc

Table 8: Per-dataset training configuration for the GraphSAGE backbone.

G.5 Complete numerical results

Tables 9–11 report full results (mean ± standard deviation) on the homophilic, heterophilic, and large-scale datasets.

Table 9: **Test accuracy on homophilic node-classification benchmarks.** Results are reported as mean and standard deviation (%) at retained-edge fractions $\rho \in \{0.3, 0.5, 0.7\}$; higher values indicate better performance.

Dataset	ρ	Reference anchors				Topology-based											Semantic-based			EDiS (Ours)
		MLP (no graph)	Spanning Tree	Full GCN	EDiS-Full	Random	Local Degree	Rank Degree	Forest Fire	SCAN	L-Spar	G-Spar	L-Sim	Spectral	*t*-Spanner	DSpar	Unified-LTH	AdaGLT	MoG	EDiS-Lite
Cora	0.3	60.14 ±2.62	80.86 ±1.17	84.26 ±0.66	84.08 ±0.83	75.00 ±0.96	*75.16* ±0.87	76.12 ±0.46	71.20 ±0.83	71.80 ±1.51	71.54 ±0.85	71.54 ±0.85	74.76 ±0.88	69.90 ±2.08	70.67 ±0.90	69.33 ±0.87	68.50 ±2.57	73.37 ±0.62	72.80 ±0.29	**77.16** ±0.98
	0.5				83.30 ±0.61	79.10 ±0.80	*80.68* ±0.56	80.60 ±0.76	80.98 ±0.58	78.32 ±0.83	74.68 ±0.82	74.68 ±0.82	**81.14** ±0.72	75.37 ±2.53	75.67 ±1.95	77.33 ±0.75	73.57 ±1.92	76.80 ±0.86	75.90 ±1.47	80.28 ±0.75
	0.7				84.28 ±0.55	81.10 ±0.72	*82.74* ±0.68	82.46 ±0.57	80.60 ±0.40	81.82 ±0.63	78.02 ±0.67	77.92 ±0.74	80.42 ±0.35	82.78 ±0.72	80.56 ±0.27	81.20 ±0.86	79.72 ±2.26	80.20 ±0.95	81.50 ±0.81	**82.88** ±0.50
CiteSeer	0.3	57.00 ±1.13	70.50 ±0.64	72.50 ±0.36	70.88 ±0.78	64.18 ±0.59	66.12 ±0.96	64.38 ±0.82	63.02 ±1.33	64.88 ±0.84	64.76 ±0.57	64.76 ±0.57	65.26 ±0.98	64.23 ±1.05	64.37 ±0.73	**66.60** ±0.80	65.40 ±1.26	64.33 ±0.90	65.70 ±1.14	*66.08* ±1.23
	0.5				70.88 ±0.47	67.34 ±0.60	*69.04* ±0.58	**69.74** ±0.33	68.60 ±1.10	68.74 ±0.59	65.80 ±0.72	65.80 ±0.72	69.42 ±0.55	66.30 ±0.22	66.90 ±0.71	68.63 ±0.52	66.23 ±0.59	66.03 ±0.54	67.73 ±0.97	68.20 ±0.61
	0.7				71.14 ±0.29	70.16 ±0.53	70.20 ±0.55	70.40 ±0.45	**71.50** ±0.48	70.08 ±0.24	69.64 ±0.79	69.66 ±0.77	70.56 ±1.18	70.34 ±0.82	70.72 ±0.61	70.44 ±0.37	69.18 ±0.95	69.46 ±0.43	70.14 ±0.99	*70.64* ±0.84
PubMed	0.3	69.82 ±1.86	80.40 ±0.99	80.76 ±0.55	80.20 ±0.46	73.86 ±0.56	76.76 ±1.41	*75.84* ±0.41	73.96 ±1.28	73.58 ±0.71	73.58 ±0.59	73.58 ±0.59	75.54 ±0.60	74.73 ±0.71	74.43 ±1.18	75.47 ±0.25	72.33 ±1.75	75.67 ±0.37	73.80 ±0.59	**77.28** ±0.57
	0.5				80.24 ±0.61	76.52 ±0.71	79.64 ±0.77	*79.56* ±0.40	78.86 ±0.51	77.10 ±0.66	74.76 ±0.72	74.76 ±0.72	79.12 ±0.66	77.53 ±0.75	77.17 ±0.56	76.73 ±0.45	73.53 ±0.45	76.83 ±0.63	76.93 ±0.53	**80.32** ±0.72
	0.7				81.02 ±0.38	76.82 ±0.55	*80.56* ±0.83	80.18 ±0.48	78.52 ±0.99	79.02 ±0.63	77.90 ±1.08	77.90 ±1.08	80.42 ±0.26	79.04 ±0.41	78.20 ±0.58	**81.00** ±0.59	77.00 ±1.77	77.94 ±1.72	76.72 ±1.62	80.62 ±0.54
Amazon-Computer	0.3	86.99 ±0.39	89.76 ±0.30	93.70 ±0.31	93.59 ±0.05	90.95 ±0.22	89.62 ±0.12	88.22 ±0.24	*93.14* ±0.09	91.78 ±0.06	91.55 ±0.20	91.07 ±0.39	92.91 ±0.18	92.20 ±0.33	90.82 ±0.27	93.29 ±0.17	85.79 ±0.41	91.31 ±0.14	91.48 ±0.46	**93.35** ±0.21
	0.5				93.75 ±0.17	92.17 ±0.30	90.59 ±0.40	89.06 ±0.08	*93.55* ±0.17	92.37 ±0.05	91.98 ±0.14	92.03 ±0.20	93.34 ±0.15	93.32 ±0.06	92.39 ±0.11	**93.75** ±0.13	88.76 ±0.55	91.45 ±0.06	92.23 ±0.24	93.60 ±0.14
	0.7				93.87 ±0.24	93.30 ±0.23	91.48 ±0.13	90.63 ±0.25	93.29 ±0.36	93.19 ±0.11	92.58 ±0.22	92.34 ±0.05	*93.58* ±0.22	93.41 ±0.17	93.51 ±0.18	**94.07** ±0.08	91.08 ±0.07	92.32 ±0.21	93.55 ±0.14	93.88 ±0.11
Amazon-Photo	0.3	92.16 ±0.24	92.46 ±0.12	95.95 ±0.14	96.14 ±0.19	95.08 ±0.19	94.20 ±0.06	93.70 ±0.31	**95.75** ±0.18	94.18 ±0.05	93.86 ±0.24	94.01 ±0.19	95.19 ±0.15	95.14 ±0.16	94.60 ±0.06	95.21 ±0.35	91.22 ±0.57	94.40 ±0.48	*95.29* ±0.09	95.60 ±0.15
	0.5				96.06 ±0.31	95.45 ±0.16	94.42 ±0.45	94.29 ±0.11	95.62 ±0.16	94.60 ±0.13	94.23 ±0.27	94.55 ±0.26	95.25 ±0.27	95.19 ±0.29	95.14 ±0.29	*95.51* ±0.11	93.27 ±0.70	94.99 ±0.30	95.45 ±0.36	**95.82** ±0.19
	0.7				95.71 ±0.11	*95.84* ±0.25	94.75 ±0.29	94.92 ±0.20	95.64 ±0.06	95.16 ±0.11	95.10 ±0.21	94.86 ±0.43	95.51 ±0.11	**96.33** ±0.32	*95.84* ±0.32	95.73 ±0.13	93.97 ±0.32	94.95 ±0.29	95.64 ±0.08	95.88 ±0.05
Coauthor-CS	0.3	95.11 ±0.05	94.68 ±0.14	95.61 ±0.36	95.98 ±0.15	94.77 ±0.03	94.93 ±0.27	94.67 ±0.14	94.86 ±0.15	95.17 ±0.14	94.87 ±0.31	94.96 ±0.12	95.13 ±0.29	94.80 ±0.21	94.68 ±0.10	94.55 ±0.18	94.80 ±0.13	*95.16* ±0.13	94.92 ±0.31	**95.72** ±0.06
	0.5				95.95 ±0.13	95.42 ±0.16	94.97 ±0.11	94.97 ±0.07	95.29 ±0.22	95.04 ±0.20	95.28 ±0.12	95.12 ±0.14	95.18 ±0.14	95.45 ±0.16	95.18 ±0.15	95.06 ±0.08	95.04 ±0.60	*95.44* ±0.20	95.41 ±0.17	**95.86** ±0.15
	0.7				95.85 ±0.05	95.57 ±0.27	95.48 ±0.01	95.63 ±0.08	95.55 ±0.11	95.60 ±0.19	*95.70* ±0.10	95.47 ±0.04	95.59 ±0.12	95.46 ±0.13	95.31 ±0.09	95.76 ±0.14	95.26 ±0.14	95.53 ±0.10	95.56 ±0.10	**95.95** ±0.11
Coauthor-Physics	0.3	96.36 ±0.01	96.94 ±0.02	97.51 ±0.07	97.45 ±0.03	96.79 ±0.13	96.55 ±0.08	96.64 ±0.02	*96.93* ±0.03	96.87 ±0.03	96.82 ±0.04	96.82 ±0.04	96.76 ±0.06	96.95 ±0.09	96.76 ±0.11	96.81 ±0.05	96.49 ±0.00	96.65 ±0.00	96.59 ±0.11	**97.06** ±0.09
	0.5				97.43 ±0.10	96.86 ±0.05	96.85 ±0.07	96.77 ±0.14	97.22 ±0.04	97.02 ±0.06	96.92 ±0.07	96.95 ±0.05	97.01 ±0.04	97.15 ±0.07	97.00 ±0.09	*97.21* ±0.12	96.74 ±0.00	96.81 ±0.00	97.01 ±0.12	**97.34** ±0.05
	0.7				97.47 ±0.03	97.21 ±0.05	97.04 ±0.07	96.77 ±0.04	*97.29* ±0.08	97.21 ±0.08	97.10 ±0.04	97.01 ±0.09	97.24 ±0.04	97.11 ±0.05	97.27 ±0.04	97.37 ±0.11	96.67 ±0.00	97.01 ±0.00	*97.29* ±0.09	**97.47** ±0.07
WikiCS	0.3	73.10 ±0.19	75.05 ±0.23	80.57 ±0.06	80.39 ±0.29	77.28 ±0.20	75.01 ±0.29	75.89 ±0.34	79.87 ±0.32	75.01 ±0.29	74.75 ±0.22	74.65 ±0.28	78.71 ±0.38	78.98 ±0.70	77.14 ±0.24	*79.58* ±0.80	75.37 ±0.58	76.39 ±0.13	77.23 ±0.32	**80.25** ±0.79
	0.5				80.33 ±0.32	78.99 ±0.09	76.59 ±0.15	76.44 ±0.30	**80.67** ±0.16	75.02 ±0.41	75.40 ±0.49	75.34 ±0.62	79.32 ±0.21	79.49 ±0.62	78.38 ±0.50	*80.13* ±0.51	76.73 ±0.57	77.85 ±0.32	78.33 ±0.66	80.19 ±0.33
	0.7				80.17 ±0.40	78.57 ±0.54	78.38 ±0.12	77.26 ±0.32	79.54 ±0.66	77.50 ±0.39	77.15 ±0.35	77.21 ±0.24	79.45 ±0.24	79.51 ±0.34	*79.75* ±0.55	**80.31** ±0.49	77.74 ±0.60	78.40 ±0.32	79.53 ±0.62	80.17 ±0.43

Each cell reports the mean above ± standard deviation. **Bold** (teal), underline (gold), and *italic* (lavender) identify the highest, second-highest, and third-highest distinct means within each row, respectively; ties share a rank. Reference anchors are excluded from ranking. EDiS-Lite and EDiS-Full use sampled-graph and full-graph evaluation, respectively.

Table 10: **Predictive performance on heterophilic node-classification benchmarks.** The metric is accuracy for Squirrel, Chameleon, Roman-Empire, and Amazon-Ratings, and ROC-AUC for Minesweeper and Questions. Results are reported as mean and standard deviation (%) at retained-edge fractions $\rho \in \{0.3, 0.5, 0.7\}$; higher values indicate better performance.

Dataset	ρ	Reference anchors				Topology-based											Semantic-based			EDiS (Ours)
		MLP (no graph)	Spanning Tree	Full GCN	EDiS-Full	Random	Local Degree	Rank Degree	Forest Fire	SCAN	L-Spar	G-Spar	L-Sim	Spectral	t-Spanner	DSpar	Unified-LTH	AdaGLT	MoG	EDiS-Lite
Squirrel	0.3	39.77 ±1.59	36.93 ±1.58	44.79 ±1.82	43.26 ±2.43	41.04 ±1.68	40.73 ±1.95	40.45 ±1.81	43.81 ±2.21	39.87 ±1.56	38.69 ±1.00	38.77 ±1.45	38.58 ±1.83	*44.31* ±1.36	42.21 ±3.29	44.39 ±2.63	39.48 ±0.49	**45.94** ±0.92	42.30 ±1.67	42.63 ±2.04
	0.5				43.89 ±2.09	42.54 ±2.39	42.49 ±1.69	41.17 ±2.36	44.61 ±1.98	39.08 ±1.64	38.92 ±1.59	38.96 ±1.62	40.07 ±1.84	44.08 ±2.02	*44.75* ±1.84	**46.83** ±1.62	39.26 ±0.38	45.42 ±1.27	41.69 ±2.58	44.50 ±1.69
	0.7				44.03 ±2.18	43.06 ±1.63	43.45 ±1.79	42.02 ±2.03	43.15 ±2.40	38.80 ±1.70	38.22 ±1.83	38.60 ±1.71	43.31 ±1.07	**45.58** ±1.32	43.11 ±2.30	44.55 ±1.55	36.66 ±2.89	43.25 ±1.78	*44.19* ±2.22	44.15 ±1.65
Chameleon	0.3	36.59 ±3.32	34.72 ±2.81	44.67 ±3.84	43.13 ±2.53	37.35 ±3.05	42.93 ±3.82	40.49 ±4.17	41.75 ±4.27	37.31 ±4.20	40.25 ±3.34	41.56 ±3.33	39.55 ±5.50	39.95 ±0.85	39.78 ±4.30	*42.22* ±2.96	33.57 ±1.11	38.15 ±2.57	38.92 ±1.13	**43.75** ±4.14
	0.5				44.14 ±4.15	39.70 ±3.51	43.58 ±4.85	42.76 ±2.99	*44.82* ±3.43	37.69 ±3.95	38.99 ±4.06	38.88 ±4.64	41.40 ±3.50	42.12 ±3.40	40.71 ±2.17	**46.14** ±0.86	34.25 ±3.09	36.67 ±1.40	42.78 ±3.18	45.64 ±2.72
	0.7				43.50 ±4.27	44.08 ±3.81	44.37 ±4.33	43.10 ±2.08	42.93 ±4.54	35.87 ±2.91	36.92 ±3.22	36.09 ±3.61	41.21 ±2.69	**45.95** ±1.83	45.47 ±3.95	**45.95** ±4.48	36.55 ±4.18	36.72 ±3.10	44.64 ±2.60	*44.88* ±4.25
Roman-Empire	0.3	66.23 ±0.07	87.36 ±0.23	90.74 ±0.03	75.00 ±0.51	67.08 ±0.23	70.57 ±0.35	69.82 ±0.10	66.47 ±0.21	**73.31** ±0.12	71.87 ±0.13	*71.72* ±0.10	71.38 ±0.20	66.94 ±0.34	67.25 ±0.10	67.23 ±0.12	59.21 ±0.42	66.15 ±0.00	70.13 ±0.38	68.58 ±0.12
	0.5				79.61 ±0.28	71.38 ±0.29	*76.52* ±0.15	75.72 ±0.28	72.66 ±0.21	76.53 ±0.41	76.17 ±0.36	76.04 ±0.14	**78.39** ±0.17	70.83 ±0.25	71.18 ±0.38	70.91 ±0.14	61.90 ±0.64	68.13 ±0.00	73.91 ±0.63	73.71 ±0.57
	0.7				86.19 ±0.53	77.92 ±0.41	83.35 ±0.22	82.73 ±0.19	78.76 ±0.17	80.49 ±0.74	79.97 ±0.27	80.46 ±0.18	82.33 ±0.25	76.26 ±0.63	78.17 ±0.19	77.26 ±0.66	70.81 ±0.37	**90.08** ±0.00	79.65 ±0.37	*83.03* ±0.32
Amazon-Ratings	0.3	50.32 ±0.24	51.73 ±0.37	53.52 ±0.42	53.16 ±0.68	49.01 ±0.20	52.03 ±0.23	*51.91* ±0.17	48.45 ±0.56	50.57 ±0.22	49.90 ±0.16	50.11 ±0.29	51.68 ±0.62	47.36 ±0.10	48.34 ±0.36	48.17 ±0.70	41.19 ±0.11	47.02 ±0.52	51.70 ±0.63	**52.31** ±0.85
	0.5				54.12 ±0.77	51.43 ±0.14	52.06 ±0.17	51.38 ±0.48	51.16 ±0.28	51.80 ±0.41	51.95 ±0.21	51.75 ±0.31	*52.16* ±0.20	50.16 ±0.25	50.59 ±0.18	50.63 ±0.33	43.36 ±0.79	48.66 ±0.22	**53.86** ±0.66	53.62 ±0.58
	0.7				53.27 ±0.84	52.26 ±0.09	52.14 ±0.69	51.59 ±0.35	52.08 ±0.32	52.34 ±0.12	52.60 ±0.20	52.49 ±0.11	52.45 ±0.26	51.40 ±0.40	*52.65* ±0.61	52.38 ±0.20	47.15 ±0.27	51.64 ±0.33	**53.28** ±0.34	53.14 ±0.27
Minesweeper	0.3	51.34 ±0.77	70.73 ±0.21	97.49 ±0.18	82.62 ±1.09	71.26 ±0.42	72.24 ±0.62	70.44 ±0.97	67.78 ±1.53	80.06 ±0.13	79.59 ±0.39	79.78 ±0.49	78.10 ±0.56	72.33 ±1.03	72.54 ±1.46	*80.01* ±0.26	69.46 ±1.16	61.93 ±1.04	**80.83** ±0.43	73.82 ±0.97
	0.5				87.54 ±0.32	80.23 ±0.11	80.12 ±0.46	78.67 ±0.76	78.94 ±0.63	90.61 ±0.12	*90.87* ±0.12	90.98 ±0.20	**91.02** ±0.31	78.34 ±0.60	79.47 ±1.41	80.79 ±0.54	73.37 ±2.39	69.93 ±1.52	82.53 ±0.43	82.21 ±0.60
	0.7				89.84 ±0.28	85.18 ±0.50	86.30 ±0.37	86.26 ±0.81	85.26 ±0.54	*90.20* ±0.33	90.43 ±0.22	**90.46** ±0.10	89.28 ±0.07	83.95 ±0.53	85.22 ±0.72	85.51 ±0.53	82.40 ±0.81	81.28 ±1.45	87.53 ±0.14	85.37 ±0.59
Questions	0.3	69.69 ±0.35	74.85 ±0.24	78.49 ±0.18	77.60 ±0.96	72.09 ±0.26	73.34 ±0.25	71.49 ±0.48	71.34 ±0.46	72.72 ±0.45	70.29 ±1.82	69.74 ±0.98	**75.83** ±0.28	74.62 ±0.45	71.82 ±1.23	72.77 ±0.47	72.42 ±0.64	72.34 ±0.33	68.90 ±1.68	*73.88* ±0.95
	0.5				78.07 ±0.13	71.92 ±0.48	77.16 ±0.22	76.28 ±0.31	73.45 ±0.96	75.36 ±0.29	70.87 ±1.22	69.46 ±0.39	**78.12** ±0.20	77.35 ±0.69	73.25 ±0.49	77.07 ±0.59	74.89 ±0.84	73.87 ±0.73	73.13 ±0.14	*77.17* ±0.71
	0.7				78.53 ±0.38	75.65 ±0.28	77.98 ±0.38	77.15 ±0.34	75.37 ±0.65	77.17 ±0.68	73.56 ±0.69	73.57 ±0.70	78.37 ±0.44	78.10 ±0.54	76.61 ±0.69	**78.47** ±0.16	75.97 ±0.29	73.36 ±0.59	76.00 ±0.24	*78.21* ±0.46

Each cell reports the mean above ± standard deviation. **Bold** (teal), underline (gold), and *italic* (lavender) identify the highest, second-highest, and third-highest distinct means within each row, respectively ties share a rank. Reference anchors are excluded from ranking. EDiS-Full is reported as a full-graph inference reference, while EDiS-Lite uses sampled-graph evaluation.

Table 11: **Predictive performance on large-scale node-classification benchmarks.** The metric is accuracy for Reddit, ogbn-products, ogbn-arxiv, and Pokec, and ROC-AUC for ogbn-proteins. Results are reported as mean and standard deviation (%) at retained-edge fractions $\rho \in \{0.1, 0.3, 0.5\}$; higher values indicate better performance.

Dataset	ρ	Reference anchors				Topology-based											Semantic-based			EDiS (Ours)
		MLP (no graph)	Spanning Tree	Full GCN	EDiS-Full	Random	Local Degree	Rank Degree	Forest Fire	SCAN	L-Spar	G-Spar	L-Sim	Spectral	t-Spanner	DSpar	Unified-LTH	AdaGLT	MoG	EDiS-Lite
Reddit	0.1	69.66 ±0.09	86.07 ±0.14	95.44 ±0.05	95.01 ±0.00	93.13 ±0.08	81.85 ±0.16	71.85 ±0.45	94.97 ±0.04	83.96 ±0.02	83.61 ±0.03	83.54 ±0.04	93.39 ±0.01	*95.06* ±0.01	93.47 ±0.00	95.09 ±0.02	91.90 ±0.00	71.45 ±0.00	85.15 ±0.07	**95.73** ±0.02
	0.3				95.08 ±0.02	94.69 ±0.02	90.82 ±0.15	77.28 ±0.04	95.41 ±0.00	90.28 ±0.01	90.06 ±0.01	90.05 ±0.01	94.35 ±0.05	95.41 ±0.03	94.72 ±0.01	*95.22* ±0.01	92.68 ±0.00	85.09 ±0.00	90.13 ±0.08	**95.61** ±0.03
	0.5				95.15 ±0.01	95.04 ±0.01	93.32 ±0.05	79.04 ±0.10	**95.50** ±0.04	92.13 ±0.01	91.84 ±0.00	91.81 ±0.03	94.82 ±0.02	95.44 ±0.05	95.27 ±0.02	95.26 ±0.01	92.96 ±0.00	89.44 ±0.00	91.61 ±0.09	*95.42* ±0.03
ogbn-products	0.1	45.46 ±0.19	67.79 ±0.09	80.67 ±0.10	80.76 ±0.00	71.93 ±0.03	73.40 ±0.08	61.09 ±0.01	73.72 ±0.13	65.85 ±0.04	65.85 ±0.08	65.76 ±0.10	73.66 ±0.05	*73.86* ±0.14	71.84 ±0.17	77.18 ±0.26	71.72 ±0.00	61.17 ±0.00	63.45 ±0.09	**77.50** ±0.00
	0.3				81.38 ±0.13	79.08 ±0.06	77.62 ±0.16	63.13 ±0.02	80.65 ±0.05	72.56 ±0.15	72.64 ±0.02	72.55 ±0.05	79.20 ±0.09	*80.00* ±0.15	79.30 ±0.01	77.43 ±0.08	75.26 ±0.00	66.74 ±0.00	68.27 ±0.05	**80.78** ±0.42
	0.5				80.76 ±0.00	80.66 ±0.09	80.12 ±0.03	65.47 ±0.17	**82.10** ±0.22	77.19 ±0.15	76.55 ±0.08	76.49 ±0.10	80.68 ±0.07	81.89 ±0.02	*80.70* ±0.11	76.90 ±0.23	77.05 ±0.00	72.57 ±0.00	70.97 ±0.17	80.51 ±0.00
ogbn-arxiv	0.1	55.60 ±0.16	67.27 ±0.12	72.83 ±0.20	71.39 ±0.12	61.64 ±0.04	61.87 ±0.05	58.47 ±0.26	*62.41* ±0.18	61.59 ±0.07	60.90 ±0.02	60.93 ±0.08	**64.56** ±0.08	62.13 ±0.19	61.80 ±0.02	54.85 ±0.08	55.53 ±0.00	55.94 ±0.00	56.25 ±0.15	64.34 ±0.07
	0.3				73.18 ±0.30	67.77 ±0.18	68.34 ±0.17	62.59 ±0.07	68.75 ±0.21	67.36 ±0.37	65.82 ±0.05	65.86 ±0.06	70.69 ±0.08	*69.88* ±0.15	67.51 ±0.09	55.19 ±0.13	56.37 ±0.00	55.93 ±0.00	56.42 ±0.16	**70.96** ±0.23
	0.5				72.82 ±0.01	70.55 ±0.42	69.74 ±0.37	63.14 ±0.12	71.10 ±0.10	70.64 ±0.14	67.68 ±0.02	67.64 ±0.05	*71.67* ±0.18	71.93 ±0.02	70.25 ±0.20	55.95 ±0.11	57.01 ±0.00	56.62 ±0.00	56.65 ±0.15	**72.50** ±0.05
ogbn-proteins	0.1	72.91 ±0.17	74.13 ±0.16	77.48 ±0.06	75.98 ±0.62	76.50 ±0.72	69.09 ±0.16	55.64 ±0.91	**78.20** ±0.14	64.55 ±0.28	64.55 ±0.21	64.43 ±0.20	76.38 ±0.48	*77.23* ±0.35	78.01 ±0.50	53.43 ±0.01	66.46 ±0.00	63.58 ±0.00	71.12 ±0.08	76.46 ±0.45
	0.3				76.56 ±0.22	76.83 ±0.08	71.10 ±0.00	66.01 ±0.78	**78.25** ±0.32	70.62 ±0.42	70.80 ±0.39	70.39 ±0.56	77.95 ±0.11	77.09 ±0.66	*77.43* ±0.12	51.94 ±1.16	67.15 ±0.00	52.31 ±0.00	72.72 ±1.75	76.43 ±0.22
	0.5				76.42 ±0.47	*77.17* ±0.92	75.21 ±0.63	68.20 ±1.01	**77.54** ±0.58	74.62 ±0.05	74.87 ±0.18	74.30 ±0.38	76.75 ±0.24	77.08 ±0.02	77.22 ±0.13	54.99 ±1.48	64.73 ±0.00	59.33 ±0.00	70.79 ±0.05	76.45 ±0.63
Pokec	0.1	62.31 ±0.02	69.36 ±0.01	86.24 ±0.04	81.05 ±0.14	68.86 ±0.01	**75.48** ±0.05	64.09 ±0.05	72.07 ±0.04	65.08 ±0.00	64.82 ±0.01	64.81 ±0.00	70.74 ±0.01	70.95 ±0.03	68.89 ±0.02	60.46 ±0.04	65.50 ±0.00	62.99 ±0.00	64.14 ±0.01	*71.92* ±0.11
	0.3				84.45 ±0.19	76.33 ±0.05	80.07 ±0.09	67.29 ±0.10	79.43 ±0.10	68.56 ±0.02	67.42 ±0.02	67.35 ±0.01	76.56 ±0.06	*79.65* ±0.07	76.38 ±0.07	69.39 ±0.03	68.23 ±0.00	64.42 ±0.00	66.94 ±0.07	**80.79** ±0.11
	0.5				85.49 ±0.14	80.61 ±0.02	82.26 ±0.08	70.64 ±0.08	*82.31* ±0.03	74.64 ±0.03	69.71 ±0.05	69.70 ±0.00	79.97 ±0.01	82.46 ±0.01	80.70 ±0.09	73.69 ±0.04	69.87 ±0.00	66.62 ±0.00	69.47 ±0.03	**84.19** ±0.06

Each cell reports the mean above ± standard deviation. **Bold** (teal), underline (gold), and *italic* (lavender) identify the highest, second-highest, and third-highest distinct means among ranked methods within each row, respectively; ties share a rank. Reference methods are excluded from ranking. MLP, Spanning Tree, and Full GCN are shared across retention ratios. EDiS-Full is reported as a full-graph inference reference at each training budget, while EDiS-Lite uses sampled-graph evaluation.

H Runtime and Memory Details

This appendix quantifies what EDiS's sparsity costs and saves in practice. It first covers the inference-time trade-off between EDiS-Lite and EDiS-Full (Figure 9), then a full end-to-end runtime and memory breakdown across retention budgets (Tables 12–14), and finally a comparison against external baselines under matched hardware (Tables 15 and 16).

H.1 Lite versus Full Evaluation Trade-off

Switching to full-graph evaluation mainly pays off on comparatively sparse graphs, where EDiS-Lite drops enough structure at tight budgets to cost real accuracy, and the full graph restores it. On already dense graphs, a sparse subgraph retains enough structure to stay competitive on its own, so evaluating on the full graph adds little or no accuracy while still paying a real evaluation-time cost, sometimes several times higher. The Lite/Full choice is therefore something to weigh per graph, based on how sparse or dense it already is, rather than a switch that pays off everywhere.

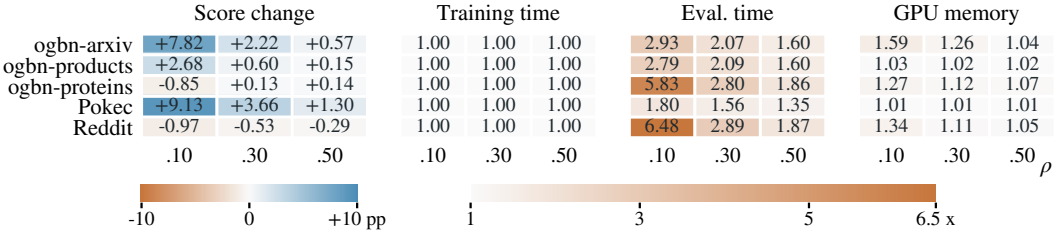


Figure 9: **Full-graph evaluation tradeoffs.** Cells show Full minus Lite score (percentage points) or Full/Lite cost ratios at retention ρ (MaxCF, cosine). The cost panels share a color scale.

Each variant is divided into construction (support generation, graph preparation, and composition), training, and evaluation, together with peak GPU and system memory. Construction and training are nearly identical between variants, since both share the same decomposition and training graphs. The gap lives entirely in evaluation, where Lite reuses the already-composed sparse graph while Full reprocesses the complete graph, shrinking as ρ grows until the two variants match at $\rho = 1$. Tables 12–14 give the full numbers behind this trade-off, in seconds and GiB, measured on a single H200 80GB GPU. This internal comparison and the cross-method comparison against baselines (Section H.2) were run on different GPU generations, so absolute times should be compared only within each table, not across the two.

	EDiS-Lite						EDiS-Full					
Dataset	Constr.	Train	Eval	Total	GPU	RAM	Constr.	Train	Eval	Total	GPU	RAM
Reddit	1400.2	126.1	49.9	1576.2	7.05	12.21	1365.5	126.0	323.6	1815.1	9.43	14.88
ogbn-products	291.4	633.1	218.2	1142.7	51.51	13.53	312.0	633.3	608.3	1553.6	53.17	13.83
ogbn-arxiv	16.2	599.2	230.6	846.0	6.79	1.79	16.5	598.9	675.9	1291.3	10.80	1.79
ogbn-proteins	355.1	26.5	10.9	392.5	4.98	8.36	356.1	26.5	63.6	446.2	6.32	8.36
Pokec	332.2	3451.2	1427.6	5211.0	49.90	5.02	334.5	3451.8	2574.1	6360.4	50.50	5.05

Table 12: Runtime and memory at $\rho = 0.1$.

	EDiS-Lite						EDiS-Full					
Dataset	Constr.	Train	Eval	Total	GPU	RAM	Constr.	Train	Eval	Total	GPU	RAM
Reddit	1547.0	257.5	112.0	1916.5	11.40	14.90	1447.4	257.4	323.4	2028.2	12.60	14.88
ogbn-products	525.8	791.4	290.7	1607.9	55.47	13.91	432.1	791.1	607.7	1830.9	56.76	13.88
ogbn-arxiv	26.9	793.3	326.4	1146.6	8.62	1.95	24.6	793.3	676.1	1494.0	10.84	1.93
ogbn-proteins	384.6	51.1	22.7	458.4	6.68	10.44	363.2	51.0	63.5	477.7	7.50	10.76
Pokec	745.7	3926.3	1648.1	6320.1	51.79	5.47	918.4	3925.6	2573.2	7417.2	52.26	5.35

Table 13: Runtime and memory at $\rho = 0.3$.

	EDiS-Lite						EDiS-Full					
Dataset	Constr.	Train	Eval	Total	GPU	RAM	Constr.	Train	Eval	Total	GPU	RAM
Reddit	1717.8	386.0	172.8	2276.6	15.95	12.40	1621.2	386.1	323.3	2330.6	16.79	14.93
ogbn-products	1422.7	984.3	380.7	2787.7	59.43	14.07	1402.5	984.2	607.8	2994.5	60.35	14.06
ogbn-arxiv	37.1	987.1	423.4	1447.6	10.44	2.05	37.3	987.0	676.0	1700.3	10.82	2.05
ogbn-proteins	449.4	75.2	34.3	558.9	8.46	8.79	444.5	75.2	63.7	583.4	9.05	8.79
Pokec	1105.3	4467.3	1899.5	7472.1	53.69	6.13	1206.8	4467.1	2573.8	8247.7	54.02	6.21

Table 14: Runtime and memory at $\rho = 0.5$.

H.2 End-to-End Runtime Against Baselines

All methods in Tables 15 and 16, EDiS-Lite and every baseline, are measured on the same single H100 80GB GPU, so the comparison is not confounded by hardware.

Sparsification time. Table 15 compares standalone sparsification time at the same retention ratio, $\delta = 0.3$, across five large-scale graphs. EDiS-Lite's reported times use each dataset's own tuned round cap $K_{\max}$ (Appendix G.4, 10–80 across these five graphs), since a deeper cap means more extraction rounds and shifts cost from composition toward precomputation (Appendix I.7), so these numbers reflect each dataset's own cap rather than a single fixed setting. EDiS-Lite's one-time extraction costs more than simple degree- or neighborhood-based heuristics (SCAN, L-Spar, G-Spar, L-Sim, Local degree), is comparable to or below Spectral, and is far below Forest Fire. This cost is paid once and reused across epochs and retention budgets (Appendix I.8), so it should be weighed against the training savings it buys, not against sparsification cost in isolation.

Table 15: Sparsification time (s) at $\delta = 0.3$ on five large-scale graphs.

Method	Reddit	ogbn-products	ogbn-arxiv	ogbn-proteins	Pokec
Random	**1.23**	**1.37**	**0.015**	**0.819**	**0.582**
Local degree	41.20	30.39	0.543	24.01	10.02
Rank degree	249.83	69.20	4.05	169.86	27.50
Forest Fire	2501.61	—	10.33	1414.05	692.14
SCAN	78.66	47.57	0.649	40.52	13.32
L-Spar	80.46	49.19	0.594	40.83	11.48
G-Spar	78.30	47.63	0.654	40.20	13.43
L-Sim	82.70	54.08	0.664	42.61	13.04
Spectral‡	387.50	379.95	5.80	286.24	108.09
Spanning tree†	36.21	17.25	0.225	2.10	3.32
EDiS-Lite	313.73	179.28	3.64	98.06	86.01

Best and second-best reported times are bold and underlined, respectively. †: spanning-tree reference with unmatched retention; —: unavailable.
‡: approximate effective resistance; using the Spielman–Srivastava JL method (Spielman & Srivastava, 2008), avoiding explicit Laplacian pseudoinversion. We use $d = \lceil 4 \ln n \rceil$ Gaussian projections with approximate-Cholesky solves (Gao et al., 2023) (10^{-2} tolerance) on the smaller graph, 64 Rademacher projections with PCG on OGBN-PRODUCTS, OGBN-PROTEINS, and POKEC, and TGT+ (Zhang et al., 2023) with 128 eigenpairs on REDDIT

End-to-end runtime. Table 16 compares end-to-end runtime at $\delta = 0.3$ over 300 epochs, splitting E_{300} into the one-time cost S and training cost T_{300}. Per epoch, EDiS-Lite trains faster than Full-GCN on all three datasets, OGBN-ARXIV (0.12 vs. 0.32 s), OGBN-PRODUCTS (1.35 vs. 3.85 s), and REDDIT (1.73 vs. 2.38 s), while matching or exceeding its accuracy (95.61% vs. 95.44% on REDDIT, Table 11). This beats Full-GCN's end-to-end runtime within 300 epochs on OGBN-ARXIV (39.5 s vs. 96.7 s) and OGBN-PRODUCTS (583.8 s vs. 1154.0 s). On REDDIT, the largest one-time cost (313.7 s) is not yet amortized at 300 epochs, so the total (832.4 s) still exceeds Full-GCN's (713.4 s). It crosses below after roughly 480 epochs, sooner with reuse across budgets or runs. DSpar, a lighter one-shot spectral sparsifier, beats EDiS-Lite on OGBN-PRODUCTS and REDDIT but not OGBN-ARXIV (70.5 s vs. 39.5 s). Its value there is accuracy and multi-budget reuse, not raw speed. MoG, AdaGLT, and Unified-LTH remain far slower across all three datasets. The marked AdaGLT and Unified-LTH configurations exceed the 3600 s run limit and are projected.

Table 16: End-to-end runtime (s) at $\delta = 0.3$ over 300 epochs. S: sparsification; T_{300}: training; $E_{300} = S + T_{300}$.

Method	Reddit			ogbn-products			ogbn-arxiv		
	S	T_{300}	E_{300}	S	T_{300}	E_{300}	S	T_{300}	E_{300}
Full-GCN	0.0	713.4	713.4	0.0	1154.0	1154.0	0.0	96.7	96.7
EDiS-Lite	313.7	518.7	832.4	179.3	404.5	583.8	3.6	35.9	**39.5**
DSpar	4.0	224.4	**228.4**	4.1	491.8	**496.0**	0.1	70.4	70.5
MoG	incl.	1432.3	1432.3	incl.	2536.4	2536.4	incl.	154.1	154.1
AdaGLT	incl.	8819.6	8819.6†	incl.	15213.3	15213.3†	incl.	916.3	916.3
Unified-LTH	5.4	7144.1	7149.5†	6.7	13161.1	13167.8†	0.1	395.9	396.0

Best and second-best E_{300} per dataset are bold and underlined, respectively. † marks AdaGLT/Unified-LTH totals exceeding the 3600 s run limit and are projected from the epochs they ran; "incl." means graph-selection cost is included in training.

I Ablations and Additional Profiles

Sections I.1–I.6 sweep the selector, score, mechanism, inference-graph, and backbone axes on the 14 homophilic/heterophilic datasets. The five large-scale graphs are profiled separately, for extraction depth (I.7) and decomposition reuse (I.8).

I.1 Extended selector ablation

The EDiS framework does not depend on which selector is used. Any of the seven arms can substitute for another with no other changes to the pipeline, and which selector performs best depends on the dataset rather than being fixed. So MaxCF is a strong default, not the choice that always wins.

Table 17 extends the main-text selector comparison to the 14 datasets. We fix the GCN backbone, cosine scores, and 30% retention. Hybrid combines MaxCF, then k-NN, then degree-normalized extraction, using the $K = 10$ round limit from Appendix F. Figure 10 extends the comparison across the full measured retention range, testing whether MaxCF's advantage as the default selector holds throughout. Selectors diverge most at low retention, where the choice of edges to keep is most consequential, and converge as $\rho \to 1$, where every selector necessarily retains the full edge set.

Dataset	MaxCF (default)	MinCF	Degree	k-NN	Spanner	Random	Hybrid
Cora	77.16±0.98	77.92±0.71	77.94±0.72	76.04±0.45	**78.02**±0.80	76.20±0.57	77.24±1.00
CiteSeer	66.08±1.23	67.26±1.05	67.38±0.88	65.34±1.00	**67.94**±0.95	67.06±0.91	66.14±1.03
PubMed	77.28±0.57	76.52±1.99	76.76±0.84	74.54±0.96	76.26±0.29	75.16±1.15	**77.76**±0.41
Amazon-Computer	**93.35**±0.21	92.98±0.11	93.06±0.18	91.54±0.33	91.76±0.53	91.57±0.22	92.07±0.22
Amazon-Photo	**95.60**±0.15	95.56±0.14	95.51±0.27	94.18±0.33	95.39±0.39	95.14±0.21	95.46±0.16
Coauthor-CS	**95.72**±0.06	95.06±0.09	95.23±0.14	95.39±0.28	95.14±0.22	95.42±0.19	95.44±0.09
Coauthor-Physics	97.06±0.09	96.84±0.06	97.03±0.09	97.00±0.08	96.70±0.08	96.66±0.08	**97.11**±0.10
WikiCS	**80.25**±0.79	77.85±0.31	79.61±0.45	78.72±0.36	77.80±0.35	77.34±0.37	78.44±0.50
Squirrel	42.63±2.04	42.33±1.27	44.05±1.84	40.59±2.42	**44.39**±1.35	41.22±1.96	43.04±1.89
Chameleon	**43.75**±4.14	40.05±3.33	40.73±3.14	39.25±3.35	42.79±2.34	38.92±4.22	41.86±2.29
Roman-Empire	68.95±0.41	**69.06**±0.58	68.52±0.46	68.84±0.22	68.91±0.29	68.37±0.73	68.87±0.59
Amazon-Ratings	52.31±0.85	52.72±0.44	52.34±0.52	51.87±0.42	51.99±0.30	51.94±0.51	**52.76**±0.31
Minesweeper†	73.82±0.97	**77.84**±0.82	75.69±0.34	74.25±1.25	75.01±1.05	75.30±1.51	73.74±1.68
Questions†	74.12±0.88	**74.73**±1.04	73.46±1.41	72.91±0.96	72.79±0.69	71.39±0.66	73.69±0.72

Table 17: Selector ablation for EDiS-Lite (GCN, cosine scores, $\rho = 0.30$). Values are mean ± standard deviation (%). †Reported as ROC-AUC rather than accuracy. Best and second-best reported means are bold and underlined, respectively.

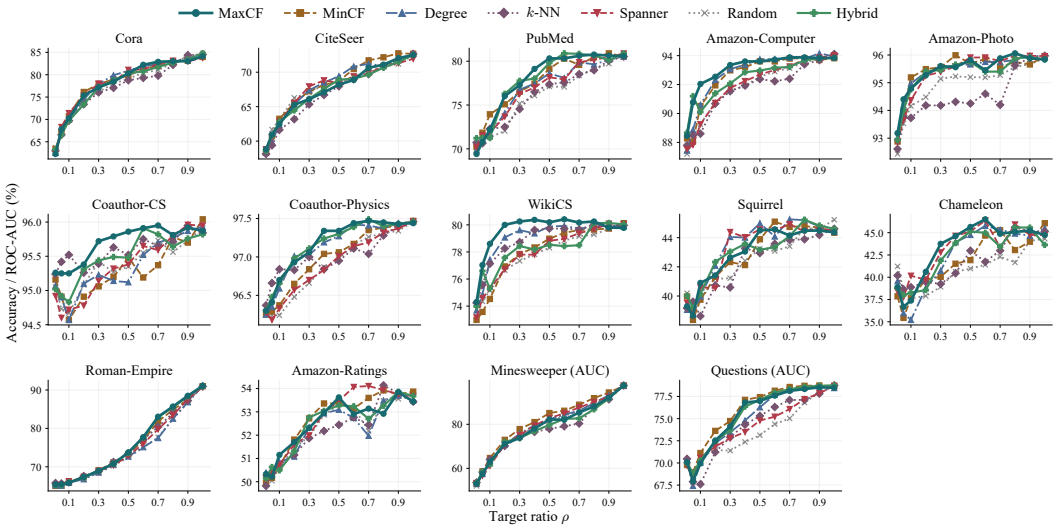


Figure 10: **Selector ablation across the full retention range.** GCN backbone, cosine scores, all 14 datasets. MaxCF (default) and Hybrid are drawn with heavier lines.

I.2 Extended edge-score ablation

The EDiS framework does not depend on which edge score is used. Any of the five scores can substitute for another with no other changes to the pipeline, and which score performs best depends on the dataset rather than being fixed. So cosine is a strong default, not the choice that always wins.

Table 18 extends the main-text edge-score comparison (Figure 5) to the 14 datasets. We fix MaxCF, the GCN backbone, and 30% retention while varying only the edge score. Four of the five (cosine, Euclidean, dot-product, and uniform) are fixed formulas, while MLP instead learns its preference from a small trained network over the two endpoint features. Figure 11 extends the comparison across the full measured retention range, testing whether cosine's advantage as the default score holds throughout. Scores diverge most at low retention, where edge weighting is most consequential, and converge as $\rho \to 1$, where every score sees the same edge set.

Dataset	Cosine (default)	Euclidean	Dot	Uniform	MLP
Cora	77.16±0.98	75.46±2.16	77.78±0.69	**77.94**±1.09	77.74±1.20
CiteSeer	66.08±1.23	**66.66**±1.04	66.18±0.84	66.38±0.54	66.54±1.10
PubMed	77.28±0.57	**77.80**±0.28	77.30±0.81	76.82±1.19	–
Amazon-Computer	93.35±0.21	92.58±0.03	**94.07**±0.13	92.91±0.08	93.06±0.11
Amazon-Photo	95.60±0.15	95.69±0.46	**96.01**±0.18	95.80±0.06	95.56±0.44
Coauthor-CS	**95.72**±0.06	95.71±0.05	95.51±0.13	95.04±0.18	95.28±0.15
Coauthor-Physics	97.06±0.09	97.09±0.02	**97.14**±0.12	**97.14**±0.05	97.06±0.06
WikiCS	**80.25**±0.79	80.24±0.67	79.81±0.29	79.01±0.41	79.21±0.39
Squirrel	42.63±2.04	**43.53**±2.17	42.45±2.98	43.21±2.59	42.28±1.18
Chameleon	**43.75**±4.14	40.65±3.24	43.38±3.52	41.31±2.65	40.21±2.85
Roman-Empire	68.95±0.41	69.18±0.20	68.80±0.15	**70.53**±0.49	68.90±0.37
Amazon-Ratings	52.31±0.85	52.45±0.88	**53.38**±0.50	52.94±0.59	52.74±0.21
Minesweeper†	73.82±0.97	74.26±0.26	74.00±0.80	76.67±0.61	**83.70**±1.57
Questions†	74.12±0.88	72.72±1.83	74.12±0.17	**74.15**±0.08	73.38±0.87

Table 18: Score-function ablation for EDiS-Lite (GCN, MaxCF selector, $\rho = 0.30$). Values are mean ± standard deviation (%). †Reported as ROC-AUC rather than accuracy. Best and second-best reported means are bold and underlined, respectively. A dash marks PubMed–MLP, which is not applicable in this implementation.

Diagnostic. Minesweeper is the outlier in Table 18. MLP reaches 83.70 ± 1.57 ROC-AUC, while the cosine default only reaches 73.82 ± 0.97. That is a 9.88-point gap, about six times the next-largest gap in the panel (Roman-Empire, 1.58 points). Figure 11 below shows this gap holds across most of the retention range, not just at $\rho = 0.30$. So on Minesweeper, the edge score matters more than the selector.

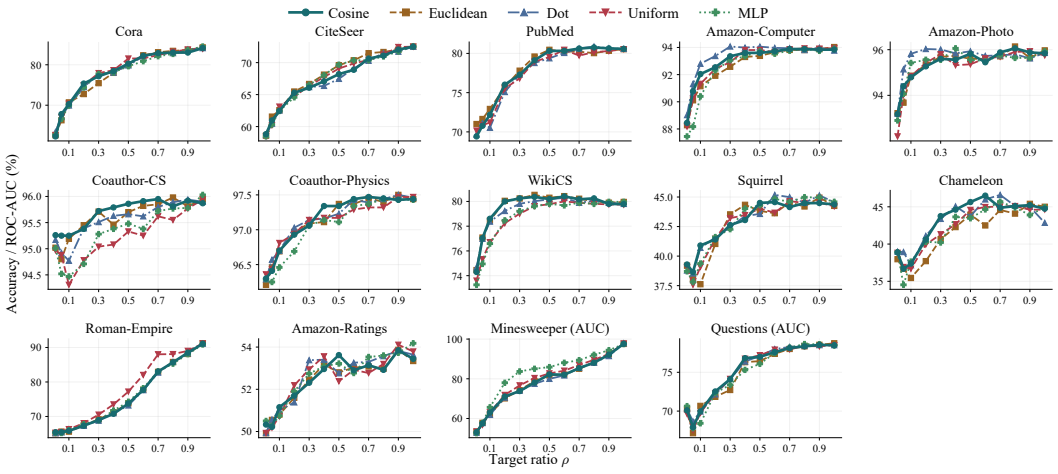


Figure 11: **Score-function ablation across the full retention range.** GCN backbone, MaxCF selector, all 14 datasets. Cosine scores (default) are drawn with a heavier line.

I.3 Extended mechanism ablation

At 30% retention, EDiS-Lite has a higher reported mean than Frozen on 13 of 14 datasets, Direct on 11, and Shuffled groups on 12. It leads all three controls on nine datasets (Table 19). The largest control advantage in this table is 0.72 points for Shuffled groups on Squirrel. These descriptive comparisons support the value of decomposition and resampling across much of the panel, while showing that neither is uniformly beneficial. The retention sweep further examines where the Frozen gap is largest.

Table 19 extends the main-text mechanism comparison (Figure 4) to the 14 datasets. We fix the GCN backbone, MaxCF selector, and cosine scores at 30% retention. Frozen reuses one sampled graph. Direct picks edges by score without forming subgraphs. Shuffled groups keeps subgraph sizes but scrambles which edges belong together, testing the contribution of group membership at matched subgraph sizes. Figure 12 extends the comparison across the full measured retention range, with each cell showing the gap between EDiS-Lite and one control, at one dataset and one ratio. This shows where each control is weakest, not just its average, and a heatmap fits three controls times twelve ratios in one view better than a line per dataset would.

Dataset	EDiS-Lite	Frozen	Direct	Shuffled groups
Cora	77.16±0.98	74.46±1.00	**77.44**±1.21	76.46±1.98
CiteSeer	66.08±1.23	65.24±0.67	**66.64**±0.79	65.96±1.05
PubMed	**77.28**±0.57	76.40±0.77	74.52±0.72	73.88±1.63
Amazon-Computer	**93.35**±0.21	92.96±0.32	91.44±0.19	91.14±0.26
Amazon-Photo	**95.60**±0.15	95.06±0.59	95.01±0.39	94.94±0.38
Coauthor-CS	**95.72**±0.06	95.59±0.05	94.97±0.22	95.05±0.33
Coauthor-Physics	97.06±0.09	**97.10**±0.11	96.74±0.09	96.84±0.08
WikiCS	**80.25**±0.79	79.98±0.35	77.47±0.49	77.54±0.56
Squirrel	42.63±2.04	41.40±2.08	43.13±1.12	**43.35**±2.37
Chameleon	**43.75**±4.14	43.73±3.19	42.14±1.98	39.79±3.50
Roman-Empire	**68.58**±0.12	67.34±0.49	68.38±0.59	68.54±0.38
Amazon-Ratings	**52.31**±0.85	50.03±0.81	52.15±0.57	51.72±0.39
Minesweeper†	73.82±0.97	69.13±0.97	73.15±0.69	**74.20**±0.44
Questions†	**74.12**±0.88	73.59±0.66	72.21±0.95	72.38±0.73

Table 19: Mechanism ablation for EDiS-Lite at $\rho = 0.30$ with GCN, MaxCF, and cosine scores. Values are mean ± standard deviation (%). †Reported as ROC-AUC rather than accuracy. Best and second-best means are bold (blue) and underlined (gray).

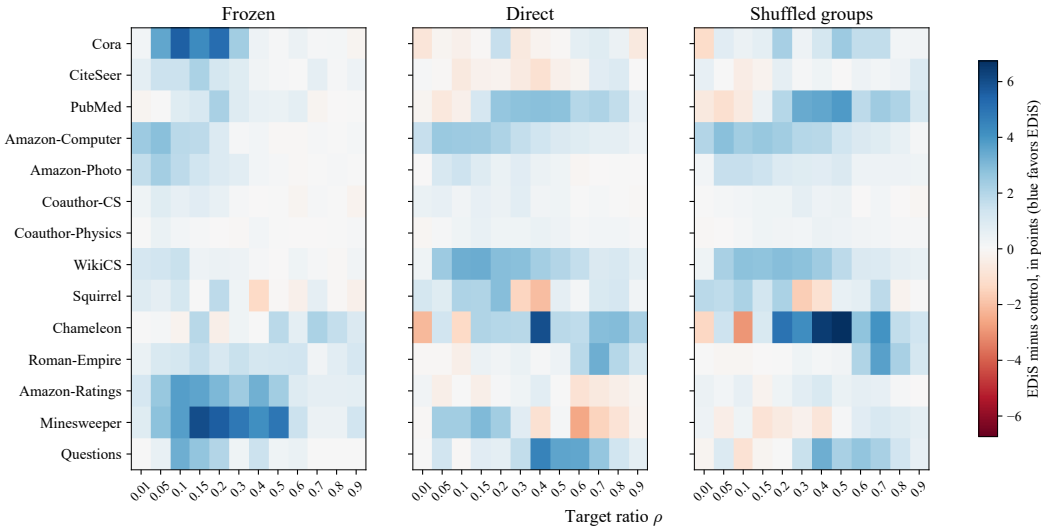


Figure 12: **Mechanism ablation across the full retention range.** GCN backbone, MaxCF selector, cosine scores, all 14 datasets. Each cell is EDiS-Lite minus the named control, in accuracy or ROC-AUC points. Blue means EDiS-Lite wins.

I.4 Extended inference-graph ablation

Evaluating on the full graph generally recovers some accuracy over EDiS-Lite, but how much depends heavily on the dataset. The gain is largest at low retention and shrinks toward zero as retention grows, since Sparse and Full become the same graph there. EDiS still evaluates on the sparse graph by default, which gives up some accuracy but keeps the method usable on very large graphs. Switching only the evaluation graph does not always close the gap to the independently-tuned Full GCN reference. EDiS-Full closely tracks that reference on several datasets (e.g., Coauthor-Physics, Amazon-Photo, WikiCS, Coauthor-CS), even slightly exceeding it on three of them, but a double-digit gap remains on Roman-Empire and Minesweeper, where the sparse training graph itself costs real accuracy.

The main text evaluates EDiS-Lite on the same sparse graph it trains on, since full-graph inference is often infeasible at the scales this method targets. Table 20 quantifies what that choice costs, comparing evaluation protocols for the 14 datasets. We fix GCN, MaxCF, and cosine scores at 30% training retention. EDiS-Lite trains and evaluates on the sparse graph. EDiS-Full trains on the same sparse graph but evaluates on the full graph, isolating what the evaluation graph contributes on its own. Full GCN trains and evaluates on the full graph throughout, as a ratio-independent reference. Figure 13 extends this comparison across the full measured retention range, with each dataset given its own y-scale since the recovered amount varies by more than $40\times$ across datasets.

Dataset	**EDiS-Lite**	**EDiS-Full**	Full GCN (ref.)	Gap (Full−Lite)
Cora	77.16±0.98	84.08±0.83	84.56±0.63	+6.92
CiteSeer	66.08±1.23	70.88±0.78	72.30±0.47	+4.80
PubMed	77.28±0.57	80.20±0.46	81.22±1.11	+2.92
Amazon-Computer	93.35±0.21	93.59±0.05	93.80±0.20	+0.24
Amazon-Photo	95.60±0.15	96.14±0.19	95.95±0.33	+0.54
Coauthor-CS	95.72±0.06	95.98±0.15	95.72±0.04	+0.26
Coauthor-Physics	97.06±0.09	97.45±0.03	97.50±0.12	+0.39
WikiCS	80.25±0.79	80.39±0.29	80.27±0.44	+0.14
Squirrel	42.63±2.04	43.26±2.43	44.77±2.36	+0.63
Chameleon	43.75±4.14	43.13±2.53	45.46±4.42	−0.62
Roman-Empire	68.58±0.12	75.00±0.51	91.05±0.13	+6.42
Amazon-Ratings	52.31±0.85	53.16±0.68	53.61±0.63	+0.85
Minesweeper†	73.82±0.97	82.62±1.09	97.18±0.45	+8.80
Questions†	74.12±0.88	77.70±0.53	78.47±0.29	+3.58

Table 20: Inference-graph ablation for EDiS at $\rho = 0.30$ (GCN, MaxCF, cosine). **EDiS-Lite** trains and evaluates on the sparse graph. **EDiS-Full** trains on the sparse graph but evaluates on the full graph. **Full GCN** trains and evaluates on the full graph. Gap is EDiS-Full minus EDiS-Lite. †ROC-AUC rather than accuracy.

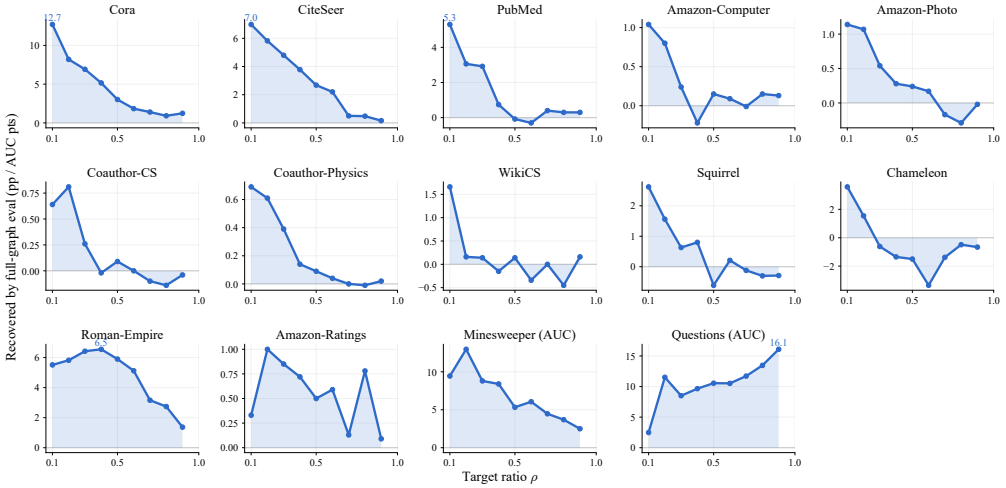


Figure 13: **How much full-graph evaluation recovers over EDiS-Lite, across the full retention range.** GCN backbone, MaxCF selector, cosine scores. One dataset per panel, each with its own y-scale since the recovered amount varies by more than $40\times$ across datasets.

I.5 Extended backbone comparison

EDiS is a framework that works with any backbone, not just GCN or GraphSAGE, and neither backbone dominates across this panel. How well EDiS performs on a given dataset depends more on how expressive that backbone is for the dataset, a property of the backbone's own architecture, than on the sparsification procedure itself. EDiS simply plugs into whichever backbone a task calls for and inherits its expressivity, rather than being tied to one. The largest gap between backbones appears on Squirrel, a few points at most, while most of the panel differs by a point or less, so the two backbones are close substitutes almost everywhere despite that occasional larger split. The stronger backbone does not track homophily in any simple way. GCN wins on some of the most heterophilic datasets in the panel (Squirrel, Chameleon) and some of the most homophilic (Cora, Coauthor-Physics) alike, so backbone choice reflects the architecture-dataset match itself rather than a graph statistic we could read off in advance.

Table 21 extends the main-text backbone comparison to the 14 datasets, at $\rho \in \{0.30, 0.50, 0.70\}$. We fix MaxCF and cosine scores and vary only the backbone. Both training and evaluation use the sparse graph. Figure 14 extends the comparison across the full measured retention range, one line per backbone per dataset.

	$\rho = 0.30$		$\rho = 0.50$		$\rho = 0.70$	
Dataset	**GCN**	**GraphSAGE**	**GCN**	**GraphSAGE**	**GCN**	**GraphSAGE**
Cora	**77.16**±0.98	75.34±0.91	**80.28**±0.75	78.90±1.03	**82.88**±0.50	81.32±0.34
CiteSeer	**66.08**±1.23	64.54±1.51	**68.20**±0.61	66.72±0.87	**70.64**±0.84	69.30±0.78
PubMed	**77.28**±0.57	75.36±1.52	**80.32**±0.72	77.52±1.03	**80.62**±0.54	77.50±1.56
Amazon-Computer	**93.35**±0.21	92.84±0.30	**93.60**±0.14	92.98±0.18	**93.88**±0.11	92.73±0.09
Amazon-Photo	95.60±0.15	**96.71**±0.08	95.82±0.19	**96.38**±0.08	95.88±0.05	**96.56**±0.20
Coauthor-CS	95.72±0.06	**96.06**±0.12	95.86±0.15	**96.11**±0.09	95.95±0.11	**96.30**±0.08
Coauthor-Physics	**97.06**±0.09	96.98±0.05	**97.34**±0.05	97.18±0.03	**97.47**±0.07	97.08±0.17
WikiCS	**80.25**±0.79	79.97±0.32	80.19±0.33	**80.76**±0.48	80.17±0.43	**80.69**±0.30
Squirrel	**42.63**±2.04	39.30±2.29	**44.50**±1.69	40.37±2.48	**44.15**±1.65	40.10±2.19
Chameleon	**43.75**±4.14	42.21±4.98	**45.64**±2.72	42.77±4.50	**44.88**±4.25	43.53±5.64
Roman-Empire	**68.95**±0.41	68.94±0.28	73.75±0.48	**73.93**±0.40	83.03±0.32	**83.35**±0.05
Amazon-Ratings	52.31±0.85	**53.19**±0.36	53.62±0.58	**54.37**±0.31	53.14±0.27	**54.58**±0.44
Minesweeper†	73.82±0.97	**74.06**±0.92	**82.21**±0.60	81.93±0.26	85.37±0.59	**85.51**±0.54
Questions†	**74.12**±0.88	72.72±0.68	**76.99**±0.54	74.87±0.86	**78.13**±0.28	75.28±0.81

Table 21: Backbone-generalization ablation for EDiS-Lite with MaxCF and cosine scores, at $\rho \in \{0.30, 0.50, 0.70\}$. Values are mean ± standard deviation (%). †Reported as ROC-AUC rather than accuracy. The higher-scoring backbone at each ratio is bold. No single backbone dominates across the panel.

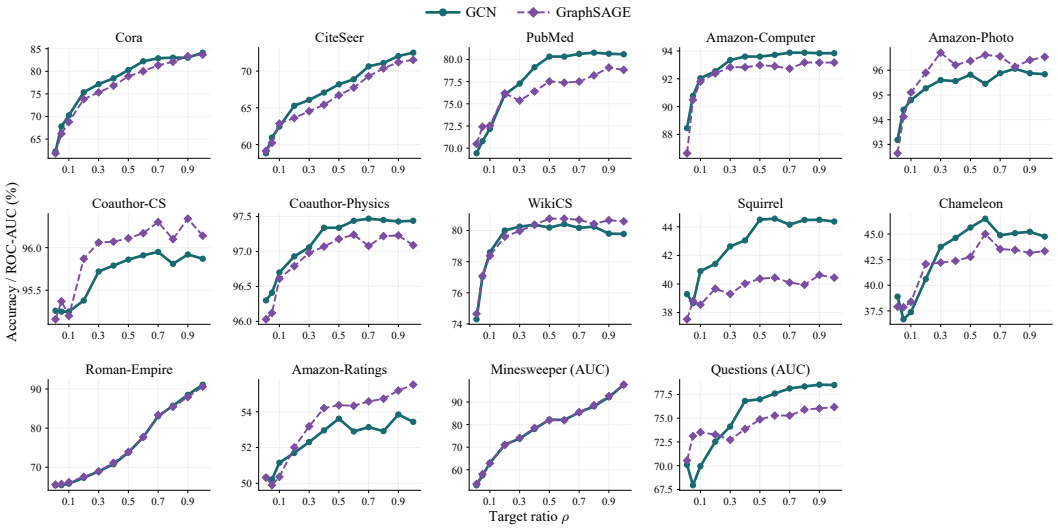


Figure 14: **Backbone comparison across the full retention range.** MaxCF selector, cosine scores, all 14 datasets. GCN is drawn heavier since it is the backbone used everywhere else in this paper.

I.6 Selector and score robustness under GraphSAGE

The same broad pattern holds under GraphSAGE as under GCN. No single selector or edge score wins on every dataset, and the defaults remain reasonable choices rather than the best possible pick everywhere, though the specific winners are not identical across backbones. Degree and k-NN never win outright under GCN anywhere in this appendix, yet under GraphSAGE Degree wins on three datasets (Coauthor-Physics, WikiCS, Minesweeper) and k-NN wins one (Roman-Empire), so which selectors are merely acceptable versus genuinely competitive is itself backbone-dependent. Minesweeper's large MLP advantage, flagged for GCN earlier in this appendix, also appears under GraphSAGE, so it is a property of the dataset rather than the backbone.

RQ5 and RQ7 each vary one axis of the pipeline while holding the other fixed, and this section checks whether the two interact. If GraphSAGE preferred a different selector or score than GCN does, RQ5's conclusion would only hold for the one backbone it was tested under. Tables 17 and 18 show that no single selector or edge score dominates under GCN. Tables 22 and 23 repeat that check under GraphSAGE, at $\rho = 0.30$. The selector table fixes cosine scores. The score table fixes MaxCF. Some selector-dataset and score-dataset pairs were not measured under this backbone and are left blank.

Dataset	MaxCF (default)	MinCF	Degree	k-NN	Spanner	Random	Hybrid
Cora	75.34±0.91	**75.42**±1.22	75.14±0.87	73.96±0.98	75.26±1.10	74.38±1.14	74.64±1.29
CiteSeer	64.54±1.51	65.62±0.35	65.18±0.92	64.54±1.33	**66.36**±0.70	65.32±0.71	64.54±0.69
PubMed	75.36±1.52	**75.62**±0.39	75.54±2.02	73.86±1.26	75.26±1.20	73.60±1.01	75.32±1.76
Amazon-Computer	92.84±0.30	**93.01**±0.23	92.62±0.10	91.37±0.48	91.62±0.15	91.37±0.57	92.18±0.29
Amazon-Photo	**96.71**±0.08	96.12±0.16	96.41±0.33	94.71±0.14	95.48±0.22	95.27±0.21	95.88±0.29
Coauthor-CS	**96.06**±0.12	95.55±0.15	95.55±0.05	95.72±0.34	95.65±0.25	95.40±0.09	95.91±0.25
Coauthor-Physics	96.98±0.05	96.75±0.13	**97.01**±0.02	96.70±0.08	96.64±0.09	96.61±0.07	96.97±0.17
WikiCS	79.97±0.32	79.38±0.53	**80.05**±0.40	79.07±0.16	78.36±0.47	78.11±0.29	78.73±0.56
Squirrel	39.30±2.29	40.38±2.01	38.97±2.02	38.96±2.12	**40.42**±1.37	39.85±1.00	39.14±1.51
Chameleon	42.21±4.98	**42.41**±4.74	40.49±6.17	41.00±4.42	40.00±6.78	40.28±4.23	41.56±7.15
Roman-Empire	68.94±0.28	69.25±0.64	69.02±0.33	**69.31**±0.19	68.88±0.85	68.88±0.28	68.98±0.55
Amazon-Ratings	**53.19**±0.36	53.09±0.26	51.96±0.33	52.46±0.35	52.16±0.53	52.96±0.35	52.76±0.33
Minesweeper†	74.06±0.92	75.00±0.60	**76.25**±0.70	73.32±0.18	75.03±0.94	75.84±1.01	73.53±1.81
Questions†	72.72±0.68	**74.65**±1.23	73.93±0.49	72.39±0.96	72.91±0.96	73.59±0.86	72.84±0.72

Table 22: Selector ablation for EDiS-Lite under GraphSAGE (cosine scores, $\rho = 0.30$). Values are mean ± standard deviation (%). †Reported as ROC-AUC rather than accuracy. Best and second-best reported means are bold and underlined, respectively.

Dataset	Cosine (default)	Euclidean	Dot	Uniform	MLP
Cora	75.34±0.91	74.82±1.42	74.82±0.37	75.48±1.65	**75.68**±1.04
CiteSeer	64.54±1.51	**65.82**±0.99	64.32±0.48	63.76±2.00	65.56±1.07
PubMed	75.36±1.52	**75.80**±2.34	75.16±1.74	75.56±1.77	–
Amazon-Computer	92.84±0.30	92.29±0.15	**93.38**±0.14	93.37±0.27	92.94±0.16
Amazon-Photo	**96.71**±0.08	96.32±0.12	96.58±0.08	96.43±0.08	96.58±0.03
Coauthor-CS	**96.06**±0.12	96.01±0.13	95.75±0.10	95.70±0.11	95.56±0.09
Coauthor-Physics	96.98±0.05	96.98±0.08	**96.99**±0.08	96.96±0.08	96.75±0.09
WikiCS	79.97±0.32	80.02±0.20	80.17±0.34	**80.39**±0.48	79.63±0.33
Squirrel	39.30±2.29	40.65±1.51	40.01±1.30	**41.07**±1.55	39.22±2.21
Chameleon	42.21±4.98	40.21±4.71	**43.56**±5.47	42.41±4.42	41.52±4.34
Roman-Empire	68.94±0.28	69.26±0.16	69.30±0.23	**70.88**±0.27	69.76±0.52
Amazon-Ratings	53.19±0.36	53.49±0.50	**53.70**±0.68	53.48±0.18	52.95±0.39
Minesweeper†	74.06±0.92	74.20±0.34	72.90±0.57	76.84±0.80	**82.82**±1.82
Questions†	72.72±0.68	71.99±1.19	73.03±0.89	72.93±0.61	**73.46**±0.96

Table 23: Score-function ablation for EDiS-Lite under GraphSAGE (MaxCF selector, $\rho = 0.30$). Values are mean ± standard deviation (%). †Reported as ROC-AUC rather than accuracy. Best and second-best reported means are bold and underlined, respectively. A dash marks PubMed–MLP, which is not applicable in this implementation.

I.7 Extraction depth on large graphs

This subsection asks how much the round cap $K_{\max}$ matters once training already uses a fixed edge-budget. $K_{\max}$ limits how many subgraphs extraction builds, not how many edges training uses, so a shallow cap could starve composition or change nothing at all.

Table 24 extends the main-text extraction-depth comparison (RQ6, Figure 6) with the actual extracted depth and the construction-time breakdown behind it. We examine $K_{\max} \in \{5, 10, 20, 40, 80, 160, 320\}$ at 30% retention on the five large graphs.

Dataset	Saturation depth	Best $K_{\max}$ (Lite)	Best $K_{\max}$ (Full)	Precompute range (s)	Composition range (s)
Pokec	67	5	80	47–86	410–438
ogbn-products	114	10	10	122–378	192–1155
ogbn-proteins†	320	10	10	75–1081	40–477
ogbn-arxiv	43	320	80	2–3	12–12
Reddit	320	20	5	135–1906	196–1232

Table 24: Extraction-depth summary for the five large graphs at $\rho = 0.30$. Saturation depth is the largest actual group count reached in the sweep. Best $K_{\max}$ is the cap with the highest reported mean under each evaluation protocol. Precompute and composition ranges span the Lite sweep from $K_{\max} = 5$ to the largest measured cap. †Reported as ROC-AUC rather than accuracy.

Extraction saturates once the residual graph runs out of edges to add, so increasing the cap does not necessarily deepen the decomposition. Arxiv, Products, and Pokec saturate well below the cap, at 43, 114, and 67 subgraphs, respectively, while Proteins and Reddit exhaust the full cap of 320. Once a dataset saturates, its stored decomposition no longer changes, so any accuracy variation observed above that point, as on Arxiv, cannot be attributed to additional extraction rounds.

Even where extraction does not saturate, deeper extraction shifts cost from composition to precomputation rather than delivering an end-to-end speedup. On Reddit, for instance, Lite precomputation time rises from cap 5 to cap 320 while composition time falls.

To separate a shifting best-observed cap from practically meaningful sensitivity, Table 25 compares a common $K_{\max} = 20$ against each dataset's highest observed test mean across the sweep, the latter reported only as a retrospective reference, not a tuning target. On ogbn-arxiv, the best cap shifts to $K = 320$ for Lite and $K = 80$ for Full, yet neither exceeds $K = 20$ by more than a third of a percentage point. On Reddit, $K = 20$ is already the best cap for Lite, leaving nothing on the table. The largest gain in the entire sweep, on ogbn-proteins, remains under one percentage point. The identity of the best cap shifts considerably more than its performance advantage does.

		EDiS-Lite		**EDiS-Full**	
Dataset	K_{common}	Best K	Δ (pt)	Best K	Δ (pt)
ogbn-arxiv	20	320	+0.27	80	+0.32
ogbn-products	20	10	+0.42	10	+0.40
ogbn-proteins†	20	10	+0.59	10	+0.83
Pokec	20	5	+0.23	80	+0.75
Reddit	20	20	0.00	5	+0.08

Table 25: Common-K vs. best observed K at $\rho = 0.3$ on the five large graphs ($K_{\text{common}} = 20$). Δ is the best cap's gain over the default, in points, and is a retrospective reference, not a tuning rule. †ROC-AUC rather than accuracy.

Observation. The extraction-depth argmax varies across datasets and inference protocols, but this overstates practical sensitivity. Across all five large graphs, the common $K_{\max} = 20$ configuration remains within one percentage point of the best observed mean for both Lite and Full. $K_{\max}$ mainly trades precomputation time against composition time, not accuracy, so it needs no precise per-dataset tuning.

I.8 Coverage, overlap, and decomposition reuse

Using the Frozen and Direct controls from the mechanism ablation at $\rho = 0.3$, we measure how training views cover the edge set over time and how one extraction serves multiple retention budgets.

Coverage and temporal overlap. Figure 15 measures cumulative edge coverage, the fraction of edges used at least once, and the Jaccard overlap between consecutive epoch graphs. Direct and Frozen mark the two extremes, one with full turnover and almost no overlap, the other with one fixed graph and no turnover at all. EDiS sits between them in these measurements, exploring nearly the whole edge set like Direct while keeping more structure in common between epochs. In these runs it covers 99–100% of edges, close to Direct, but its Jaccard overlap (0.22–0.34) is roughly double Direct's (0.17–0.18), still well short of Frozen's trivial 1.0.

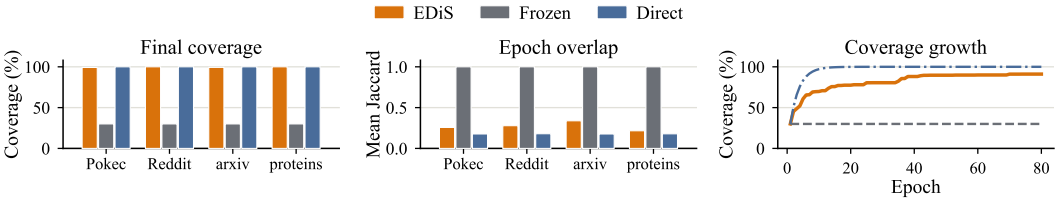


Figure 15: **Coverage and temporal overlap at $\rho = 0.3$.** Cumulative edge coverage (left), mean consecutive-epoch Jaccard (middle), and coverage over epochs for Pokec (right). Direct maximizes turnover, Frozen fixes one graph, and EDiS lies between them, near-complete coverage with greater temporal persistence than Direct.

EDiS does not maximize raw turnover. Direct explores at least as broadly, but exploration alone is not sufficient. EDiS exceeds Direct on 11 of the 14 datasets (Appendix I.3, $\rho = 0.3$), and Shuffled groups is worse despite matching EDiS's subgraph sizes, so how edges are organized into subgraphs matters, not just how many are seen. This analysis measures turnover from subgraph resampling and trimming together and does not isolate the residual pool's contribution. That separation is left to future work.

Cost of dynamic composition. Direct pays a repeated cost because it resamples from the full edge set every epoch, whereas EDiS draws from a decomposition computed once. Training, evaluation, and peak GPU memory stay essentially unchanged across all three methods, isolating the difference to graph generation alone. There, EDiS costs roughly half of Direct's per-epoch composition time (1196 ms against 2513 ms) while still varying the graph every epoch. Frozen is cheaper still, so EDiS costs more than a fixed graph but less than per-epoch resampling.

Decomposition reuse across edge-budgets. EDiS computes its decomposition independently of the edge-budget, so once cached it serves multiple ρ without rerunning extraction (Figure 16). Rebuilding separately for each of q edge-budgets grows construction cost linearly with q. EDiS pays it once, cutting cumulative cost by $5\times$ over five independent rebuilds.

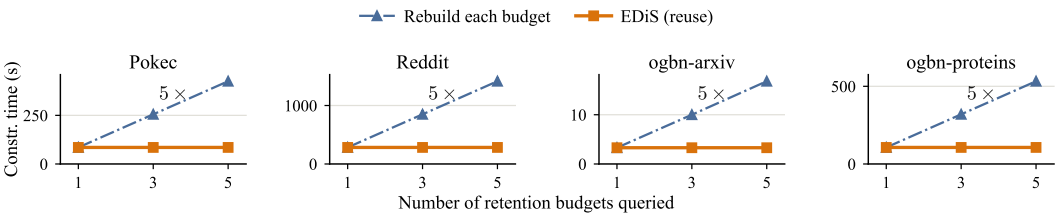


Figure 16: **Amortization of decomposition across edge-budgets.** EDiS reuses one cached decomposition. The rebuild-each-edge-budget curve is the cumulative cost of repeating extraction per edge-budget, growing linearly, whereas EDiS pays it once.

Observation. EDiS is neither a fixed sparsifier nor a maximizer of random edge turnover. In the profiled runs it covers nearly all edges over training, keeps more overlap between consecutive epochs than Direct, and costs less per epoch to compose. With the Frozen, Direct, and Shuffled groups controls, performance depends not only on how many distinct edges are seen but on how changing sparse views are organized and reused.

J Additional Experiments

J.1 Comparison with SpanGNN and GERN

Table 4 (Appendix B) compares EDiS against AGS-GNN (Das et al., 2024), SpanGNN (Gu et al., 2024), and GERN (Bonchi et al., 2025) qualitatively, by budget control, reusability, multi-structure exposure, and edge-disjointness. Here we compare them quantitatively.

AGS-GNN (Das et al., 2024) samples edges by a combination of attribute similarity and diversity, selecting diverse yet locally similar neighborhoods through greedy submodular function maximization. Its sampling objective differs from EDiS's score-based composition, and its published version provides no analysis of edge-disjoint or reusable structural properties. We attempted to include AGS-GNN under the same matched-budget protocol as our other 17 baselines, but its submodular diversity step was extremely slow and failed to complete on our evaluation hardware. AGS-GNN's own paper notes that this precomputation is embarrassingly parallel and relies on scaling up the number of worker threads to run in practice, a requirement outside the modest budget we allocate uniformly across all 19 datasets and three retention ratios. We omit AGS-GNN from Table 26.

SpanGNN (Gu et al., 2024) instead grows a single spanning subgraph incrementally across epochs to reduce memory during training, rather than composing a fresh sparse graph from reusable structure each epoch. We adapt SpanGNN to our training framework and compare it against EDiS under matched training conditions.

GERN (Bonchi et al., 2025) instead trains a GNN over a sequence of random spanning trees, each converted into a path graph and used as that epoch's sparse training structure, so it exposes the model to many cheap, independently resampled sparse views rather than a persistent, budget-controlled decomposition. We adapt GERN to our training framework, running it with the same tuned GCN backbone, optimizer, training protocol, and dataset splits as EDiS, isolating the sparsification and composition mechanism from incidental training differences.

Both methods are **evaluated under full-graph inference**, so this table reads against **EDiS-Full**, not EDiS-Lite, on the five datasets with SpanGNN and GERN results. Both reuse a sequence of sparse structures across epochs, as EDiS does, but neither separates one-time decomposition from per-epoch composition (Section 2).

	Cora			CiteSeer			PubMed			WikiCS			Amazon-Ratings		
Method	0.3	0.5	0.7	0.3	0.5	0.7	0.3	0.5	0.7	0.3	0.5	0.7	0.3	0.5	0.7
SpanGNN	74.60 ±0.76	77.74 ±1.19	77.98 ±1.66	60.60 ±2.20	64.52 ±1.55	62.82 ±0.64	73.14 ±0.55	75.66 ±0.89	76.58 ±0.87	78.70 ±0.22	78.51 ±0.06	78.56 ±0.31	51.05 ±0.48	51.35 ±0.28	51.45 ±0.50
GERN	80.46 ±0.72	80.68 ±0.57	81.10 ±0.97	68.22 ±1.24	69.34 ±1.06	70.06 ±0.86	75.88 ±1.21	76.00 ±1.33	75.58 ±1.29	80.37 ±0.23	**80.41** ±0.14	**80.42** ±0.19	46.33 ±0.19	49.35 ±0.31	51.04 ±0.60
EDiS-Full	**84.08** ±0.83	**83.30** ±0.61	**84.28** ±0.55	**70.88** ±0.78	**70.88** ±0.47	**71.14** ±0.29	**80.20** ±0.46	**80.24** ±0.61	**81.02** ±0.38	**80.39** ±0.29	80.33 ±0.32	80.17 ±0.40	**53.16** ±0.68	**54.12** ±0.77	**53.27** ±0.84

Table 26: **Accuracy (%) under full-graph inference, EDiS-Full vs. SpanGNN and GERN.** Mean above, standard deviation below, at retained-edge fraction ρ (columns), by method (rows).

J.2 Comparison with MoG under GraphSAGE

MoG (Zhang et al., 2025) mixes several sparsification experts, each favoring a different structural criterion, into one adaptive sparse graph. EDiS instead decomposes the graph once and composes each epoch's graph from that cached decomposition. MoG is one of the 17 baselines in Section 4 under our own GCN protocol. Here we use MoG's own published GraphSAGE results as an additional reference and run EDiS-Lite under the same backbone, datasets, and evaluation.

	ogbn-arxiv (Acc.)		ogbn-proteins (ROC-AUC)		ogbn-products (Acc.)	
Method	ρ=0.30	ρ=0.50	ρ=0.30	ρ=0.50	ρ=0.30	ρ=0.50
MoG (Zhang et al., 2025)	67.31	69.06	76.12	76.46	74.91	76.98
EDiS-Lite	**70.35**	**72.32**	**79.02**	**78.81**	**79.68**	**80.14**

Table 27: **MoG's published GraphSAGE results vs. EDiS-Lite.** Sparsity (70%, 50% removed) is converted to $\rho = 1 -$ sparsity. Bold marks the higher value per column.

K LIMITATIONS

Our experiments focus on node classification, so the benefits of reusable edge-disjoint decompositions for graph- and link-level prediction remain to be established. Our default edge score, residual selector, and extraction-depth cap all come from an empirical sweep rather than a principled per-dataset selection rule, and no single choice dominates across datasets, so automatic tuning of any of the three remains open. The composed graph is not guaranteed to remain connected at every fixed edge budget, and our default trimming rule favors total cached weight over expected retained score, a choice we did not test against that alternative empirically. `EDiS` also assumes the full graph is available before extraction begins, so inductive or streaming settings where edges arrive afterward are not addressed. Finally, our empirical study focuses on GCN and GraphSAGE backbones, and broader architectural validation remains future work.