MOMENT-GUIDED EDGE SAMPLING

Anonymous authors
Paper under double-blind review

ABSTRACT

Edge sampling makes local decisions to achieve graph-level objectives, such as preserving structural properties. This raises a fundamental challenge: *how can the global structural impact of removing an individual edge be quantified and controlled?* We address this challenge with a *moment-guided edge sampling framework* that summarizes global graph structure using spectral moments of the random-walk transition matrix and quantifies each edge removal through its induced moment changes. To compute moment changes efficiently, we develop two complementary methods: a combinatorial method with closed-form updates for low-order moments, and a low-rank method that reduces computation based on the endpoints of removed edges, supporting arbitrary moment orders as well as batched removals. For single-edge removal using fixed number of moments, the low-rank method reduces the cost from $O(mn)$ to $O(m)$, while the combinatorial method evaluates low-order changes in constant time given maintained local statistics. These moment changes provide **interpretable structural signatures** of local edge motifs that aggregate into graph-level fingerprints. This structural meaning motivates us to ask whether preserving moments also preserves the graph properties. We further derive and validate that moment-preserving sampling can **retain related structural properties**, including triangle-weighted clustering coefficient. These structural insights enable **analysis and improvement of graph learning**: different edge structures have distinct effects on supervised node classification, while moment-guided augmentation is competitive for graph contrastive learning. Together, these findings establish moments as an interpretable and controllable bridge from local edge edits to global graph structure and learning.

1 INTRODUCTION

Graph sampling is fundamental to graph mining and graph learning. In graph sparsification, it reduces graph size while preserving structural properties such as spectra and clustering coefficients (Leskovec & Faloutsos, 2006; Ahmed et al., 2013; Spielman & Srivastava, 2008). In supervised GNNs, sampling can regularize training, mitigate overfitting and over-smoothing (Rong et al., 2019), and improve scalability (Hamilton et al., 2017; Chiang et al., 2019; Zeng et al., 2019). In unsupervised GNNs, contrastive learning methods commonly use sampling to construct augmented graph views (You et al., 2020; Velickovic et al., 2018; Zhu et al., 2020). Across these settings, edges are often the basic sampling units, while the objectives are defined at the graph level, such as preserving structural properties. Targeted edge sampling therefore *requires a way to relate individual edges to global graph structure*. However, an edge cannot be characterized independently of its surroundings: its structural role depends on nearby edges, and its contribution to global graph properties is difficult to measure directly. What is needed is an *edge-level quantity that is locally computable yet directly reflects how an edit changes global structure*. Without such a connection, many commonly used sampling methods rely on adhocs strategies such as uniform edge dropping or random-walk sampling (Leskovec & Faloutsos, 2006; Zeng et al., 2019), rather than explicitly controlling the structural information induced by sampling.

Spectral moments of the random-walk transition matrix provide a natural bridge between local edge edits (i.e., edge additions or removals) and global graph structure. The k-th moment is the average k-step return probability over all nodes, or equivalently, a normalized sum of weighted length-k closed walks. This connection is useful in two ways. (1) *Moments summarize global graph structure*. A set of finite moments can maps each graph to a point in a moment space, providing a compact representation of its structure. Graphs with similar structural characteristics tend to occupy nearby regions of this space (Jin & Zafarani, 2020). (2) *Moment changes connect individual edge edits*

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.

to this global structure representation. Adding or removing an edge changes the weighted closed walks in the graph and therefore moves its point in the moment space. This displacement is captured by the edge-wise moment change. By computing these changes for candidate edge edits, we can select local edge edits that move the graph toward a desired target point, turning local modifications into controllable changes of global graph structure.

Present work. Building on this view, we develop a moment-guided edge sampling framework that iteratively selects edge edits based on their moment changes to move the graph toward a target moment profile. The main challenge is to compute these edge-wise moment changes efficiently. We first develop a *combinatorial method* that computes low-order moment changes by tracking affected closed walks, using maintained statistics to achieve $O(1)$ time per candidate. However, this approach becomes impractical at higher orders, as the number and variety of affected closed-walk patterns grow rapidly and make explicit tracking increasingly complex. To overcome this limitation, we develop a *low-rank method* that exploits *locality* and *cyclic trace invariance* to compress affected walks into small endpoint-to-endpoint matrices. The low-rank method supports arbitrary moment orders and batched edits, while reducing single-edge computation by a factor of n over direct recomputation. We further characterize how moment changes interact across edge edits, providing exact locality and upper bounds on pairwise interactions for arbitrary moment orders in Appendix B. Given these moment changes, we specify a target moment profile and score each candidate edge by how its removal moves the current graph toward that target, thereby linking individual edge removals to controllable changes in global graph structure. Our contributions are as follows:

- We propose an **efficient moment-guided edge sampling framework** that steers graphs toward target moment profiles. Our combinatorial method computes exact low-order changes in $O(1)$ time per candidate, while our low-rank method support arbitrary orders and batched edits with an n-fold reduction over recomputation for single-edge edits.
- We show that edge-wise moment changes provide **interpretable structural signatures**, characterizing edge motifs and forming graph-level fingerprints. We derive connections between moments and structural properties, including triangle-weighted clustering coefficient, and validate that moment-preserving sampling can **retain structural properties**.
- We explore the potential of moment-guided sampling through case studies on supervised node classification and graph contrastive learning. Moment changes reveal how different structural edge types affect supervised performance, while moment-guided augmentation achieves competitive results in contrastive learning. These results suggest that moment-guided structural control can support **analysis and improvement of graph learning**.

Code is available in an anonymous repository.

2 PRELIMINARIES

Random Walk Transition Matrix. For an undirected graph $G = (V, E)$ with $|V| = n$ nodes and $|E| = m$ edges, let $A \in \mathbb{R}^{n\times n}$ denotes its adjacency matrix, where $A_{ij} = 1$ if $(i, j) \in E$ and $A_{ij} = 0$ otherwise. Let $D \in \mathbb{R}^{n\times n}$ be the diagonal degree matrix, with $D_{ii} = \sum_{j=1}^{n} A_{ji}$. The random-walk transition matrix is $P = AD^{-1}$, where P_{ij} is the probability of moving from node j to node i in one step. Since P is similar to the symmetric matrix $P_{\text{sym}} = D^{-1/2}AD^{-1/2}$, its eigenvalues are real and lie in $[-1, 1]$.

Spectral Moments. Let $\lambda_1, \ldots, \lambda_n$ be the eigenvalues of P. The k-th spectral moment of G is $m_k = \frac{1}{n}\sum_{i=1}^{n} \lambda_i^k = \frac{1}{n}\text{Tr}(P^k)$. The diagonal entry P_{ii}^k is the probability that a k-step random walk starting at node i returns to i. Thus, m_k is the average k-step return probability over all nodes, or equivalently, a normalized sum of weighted length-k closed walks. For example, $m_2 = \frac{1}{n}\text{Tr}(P^2) = \frac{2}{n}\sum_{\{i,j\}\in E} \frac{1}{d_i d_j}$, $m_3 = \frac{1}{n}\text{Tr}(P^3) = \frac{6}{n}\sum_{\{i,j,k\}\in \triangle} \frac{1}{d_i d_j d_k}$, where d_i is the degree of node i and $\triangle$ denotes the set of unordered triangles; $m_1 = 0$ for graphs without self-loops. Spectral moments are closely connected to graph diffusion and structural properties. For example, the heat trace (Tsitsulin et al., 2018) and personalized PageRank trace (Page, 1998) can be expressed as weighted sums of moments (Appendix A) while moments are also related to spectral and structural graph properties (Preciado et al., 2013; Cohen-Steiner et al., 2018). We mainly use m_2 and m_3 for interpretation and visualization, while our method supports arbitrary moment orders.

3 MOMENT-GUIDED EDGE SAMPLING

Let $\mathcal{K} = \{k_1, \ldots, k_p\}$ denote the selected moment orders, and let $\mathbf{m} = (m_k)_{k\in\mathcal{K}}$ and $\mathbf{m}^* = (m_k^*)_{k\in\mathcal{K}}$ denote the current and target moment profiles. Our goal is to select edge edits that move $\mathbf{m}$ toward $\mathbf{m}^*$, where the target may preserve the original moments or specify a desired structural change. We represent an edge edit as $\epsilon = (o, u, v)$, where $o \in \{\text{ADD}, \text{DELETE}\}$ specifies whether the edge (u, v) is added or deleted. Let G^ϵ and P^ϵ denote the graph and random-walk transition matrix after applying ϵ. The induced change in the k-th moment is

$$\Delta m_k(\epsilon) = m_k(G^\epsilon) - m_k(G) = \frac{1}{n}\left[\mathrm{Tr}\left((P^\epsilon)^k\right) - \mathrm{Tr}(P^k)\right]. \tag{1}$$

Thus, $\Delta\mathbf{m}(\epsilon)$ describes how a local edge edit moves the graph in moment space. By evaluating these changes for candidate edits, we can select operations that move the current moment profile toward the target. The main challenge is to compute these edge-wise moment changes efficiently. Direct recomputation requires repeatedly evaluating matrix powers for many candidate edits and is prohibitively expensive on large graphs. We address this challenge from two complementary perspectives. The *combinatorial view* (Section 3.1) directly tracks the local closed-walk changes induced by an edge edit, such as destroyed, created, or reweighted triangles, yielding exact closed-form updates for low-order moments. However, explicit enumeration becomes increasingly difficult at higher orders as the number of closed-walk patterns grows rapidly. The *algebraic low-rank view* (Section 3.2) avoids explicit enumeration by exploiting *locality* and *cyclic trace invariance*. It represents edge edits as low-rank updates to P and compresses affected closed walks into small endpoint-to-endpoint matrices, supporting arbitrary moment orders and batched edits. For a single-edge edit, this reduces the cost from $O(Kmn)$ under direct recomputation to $O(Km)$. Finally, Section 3.3 uses these moment changes to iteratively select edits that move the graph toward the target profile.

3.1 COMBINATORIAL MOMENT CHANGE

The key idea is to *express moments as weighted sums of closed-walk structures*. We focus on m_2 and m_3 because they capture important degree- and triangle-related structure, as discussed in Section 5. Higher-order moments involve increasingly many closed-walk patterns, making explicit enumeration difficult. We derive single-edge deletion here, with addition formulas in Appendix B.2. Define

$$S_2 = \sum_{\{i,j\}\in E} \frac{1}{d_i d_j}, \qquad S_3 = \sum_{\{i,j,k\}\in\triangle} \frac{1}{d_i d_j d_k}. \tag{2}$$

where $\triangle$ denotes the set of unordered triangles. Since $m_2 = \frac{2}{n}S_2$ and $m_3 = \frac{6}{n}S_3$, computing their changes reduces to updating S_2 and S_3.

To avoid rescanning the entire graph after each edge edit, we maintain three local statistics:

$$W_i = \sum_{j\in\mathcal{N}(i)} \frac{1}{d_j}, \qquad M_{ij} = \sum_{k\in\mathcal{N}(i)\cap\mathcal{N}(j)} \frac{1}{d_k}, \qquad H_i = \sum_{\substack{\{a,b\}\subseteq\mathcal{N}(i)\\ \{a,b\}\in E}} \frac{1}{d_a d_b}. \tag{3}$$

Here, W_i summarizes the inverse-degree mass of the neighbors of i, M_{ij} the weighted common-neighbor mass of i and j, and H_i the weighted triangle mass around i. All quantities are evaluated before the edge edit. Once maintained, they allow the moment changes to be computed using only local information around the edited edge.

Edge deletion induced $\Delta m2$. Consider deleting an edge $(u, v) \in E$. The deletion removes the contribution of (u, v) from S_2 and reweights the surviving edges incident to u and v because their degrees decrease. For example, when $d_u > 1$, after d_u changes to $d_u - 1$, each surviving edge (u, k) with $k \neq v$ gains $\frac{1}{(d_u-1)d_k} - \frac{1}{d_u d_k} = \frac{1}{d_k}\cdot\frac{1}{d_u(d_u-1)}$. When $d_u = 1$, removing (u, v) isolates u, the corresponding reweighting contribution is zero. Summing these changes over both endpoints and subtracting the removed-edge contribution gives

$$\phi_x(y) = \begin{cases} \dfrac{W_x - d_y^{-1}}{d_x(d_x-1)}, & d_x > 1, \\ 0, & d_x = 1, \end{cases} \qquad \Delta S_2^{\mathrm{del}}(u, v) = \underbrace{\phi_u(v) + \phi_v(u)}_{\text{reweighted surviving edges}} - \underbrace{\frac{1}{d_u d_v}}_{\text{destroyed edge}}. \tag{4}$$

Edge deletion induced $\Delta m3$. Deleting $(u, v) \in E$ similarly has two effects on S_3: it destroys all triangles containing (u, v) and reweights the surviving triangles incident to u or v. For endpoint u, the total triangle mass before deletion is H_u, while the mass of triangles containing (u, v) is M_{uv}/d_v. Thus, when $d_u > 1$, the surviving triangle mass around u is $H_u - M_{uv}/d_v$, and its reweighting contributes $\left(\frac{1}{d_u-1} - \frac{1}{d_u}\right)\left(H_u - \frac{M_{uv}}{d_v}\right)$. When $d_u = 1$, removing (u, v) isolates u, so no surviving triangles remain and the corresponding reweighting contribution is zero. Combining the reweighted surviving triangles at both endpoints with the destroyed triangles gives

$$\psi_x(y) = \begin{cases} \dfrac{H_x - M_{xy}/d_y}{d_x(d_x-1)}, & d_x > 1, \\ 0, & d_x = 1, \end{cases} \qquad \Delta S_3^{\mathrm{del}}(u, v) = \underbrace{\psi_u(v) + \psi_v(u)}_{\text{reweighted surviving triangles}} - \underbrace{\frac{M_{uv}}{d_u d_v}}_{\text{destroyed triangles}}. \tag{5}$$

The corresponding moment changes are

$$\Delta m_2 = \frac{2}{n}\Delta S_2^{\mathrm{del}}(u, v), \qquad \Delta m_3 = \frac{6}{n}\Delta S_3^{\mathrm{del}}(u, v). \tag{6}$$

Once d_i, W_i, M_{ij}, and H_i are maintained, evaluating Δm_2 and Δm_3 for a candidate edge requires only constant-time lookups and arithmetic. However, extending this combinatorial approach to higher-order moments requires tracking an increasing variety of affected closed-walk patterns. This motivates the low-rank formulation in Section 3.2.

3.2 Low-rank Moment Change

To handle higher-order moments without explicitly tracking increasingly many closed-walk patterns, we derive an exact low-rank formulation that supports *arbitrary moment orders* and *batched edge edits*. We first represent the edits as a low-rank change to the transition matrix, then compress the resulting moment changes into small matrices over the edited endpoints.

Locality: edge edits form a low-rank update. Let $\mathcal{B}$ be a batch of edge edits, and let G' and P' denote the graph and transition matrix after applying all edits in $\mathcal{B}$. Only the transition columns corresponding to edited endpoints can change. We collect these nodes into the touched endpoint set

$$T = \{i \in V : i \text{ is an endpoint of an edit in } \mathcal{B}\}, \qquad r = |T| \le 2|\mathcal{B}|. \tag{7}$$

Write $T = \{i_1, \dots, i_r\}$. For each touched node i_a, let $p_a = P_{:i_a}$ and $p'_a = P'_{:i_a}$ denote its transition columns before and after the edits, and define $c_a = p'_a - p_a$. Stacking these columns gives $C = [c_1, \dots, c_r] \in \mathbb{R}^{n\times r}$ and $R = [e_{i_1}, \dots, e_{i_r}] \in \mathbb{R}^{n\times r}$, where e_i is the i-th standard basis vector. The updated transition matrix can therefore be written exactly as

$$P' = P + CR^\top. \tag{8}$$

Thus, a batch of edge edits induces a rank-r update to transition matrix P, where r depends only on the number of touched endpoints.

Cyclic trace invariance: affected closed walks are anchored at touched endpoints. Let $Q = CR^\top \in \mathbb{R}^{n\times n}$. Expanding $(P + Q)^k$ and canceling the all-P term gives

$$\Delta T_k = n\Delta m_k = \mathrm{Tr}\left((P+Q)^k\right) - \mathrm{Tr}(P^k) = \sum_{w \in \{P,Q\}^k \setminus \{P^k\}} \mathrm{Tr}(w). \tag{9}$$

Each word w contains at least one Q and represents a class of affected closed walks. Intuitively, a Q factor marks a position in the closed walk where the transition is changed at a touched endpoint, while a P factor represents an unchanged transition. These affected closed walks can therefore be characterized by two factors: (1) how many touched-endpoint occurrences they contain, and (2) where these occurrences appear along the closed walk. These correspond to the number and positions of the Q factors in w.

Let ℓ denote the number of Q factors. Grouping the affected closed walks by ℓ gives

$$\Delta T_k = \sum_{\ell=1}^{k} \sum_{\substack{w \in \{P,Q\}^k \\ w \text{ contains } \ell \text{ copies of } Q}} \mathrm{Tr}(w). \tag{10}$$

We next describe the positions of these ℓ touched-endpoint occurrences. By cyclic invariance of the trace, each closed walk can be rotated so that it starts from a Q factor. It can then be written as

$$\mathrm{Tr}\left(P^{a_1}QP^{a_2}Q\cdots P^{a_\ell}Q\right), \tag{11}$$

where a_j is the number of unchanged transitions between two consecutive touched-endpoint occurrences. Since the closed walk has total length k, with ℓ positions occupied by Q, these segment lengths satisfy $a_1 + \cdots + a_\ell = k - \ell$. Thus, all possible positions of the ℓ touched-endpoint occurrences are represented by

$$\mathcal{A}_{k,\ell} = \left\{ (a_1, \ldots, a_\ell) \in \mathbb{Z}_{\geq 0}^{\ell} : \sum_{j=1}^{\ell} a_j = k - \ell \right\}. \tag{12}$$

Cyclic rotation also changes the counting multiplicity. In the original trace expansion, a length-k closed walk can be read from any of its k positions. After anchoring the walk at a touched-endpoint occurrence, only its ℓ Q positions can serve as starting points. Therefore, each anchored representation is weighted by k/ℓ, giving

$$\Delta T_k = \sum_{\ell=1}^{k} \frac{k}{\ell} \sum_{(a_1,\ldots,a_\ell)\in\mathcal{A}_{k,\ell}} \mathrm{Tr}\left(P^{a_1}QP^{a_2}Q\cdots P^{a_\ell}Q\right). \tag{13}$$

Endpoint compression: affected walks become small matrices product. The representation above still involves the full $n \times n$ matrices P and Q. Since $Q = CR^\top$ is supported only on touched endpoints, we can further compress each walk segment between two consecutive Q factors to these endpoints. Substituting $Q = CR^\top$ into Eq. 13 and applying cyclic trace invariance gives

$$\mathrm{Tr}\left(P^{a_1}CR^\top P^{a_2}CR^\top\cdots P^{a_\ell}CR^\top\right) = \mathrm{Tr}\left[(R^\top P^{a_1}C)(R^\top P^{a_2}C)\cdots(R^\top P^{a_\ell}C)\right]. \tag{14}$$

We therefore define

$$H_t = R^\top P^t C \in \mathbb{R}^{r\times r} \tag{15}$$

Each H_t summarizes an edit-induced transition at one touched endpoint followed by t unchanged transitions that end at another touched endpoint. Thus, a closed walk with segment lengths $(a_1, \ldots, a_\ell)$ is compressed into the product $H_{a_1}H_{a_2}\cdots H_{a_\ell}$. Eq. 13 therefore becomes

$$\Delta T_k = \sum_{\ell=1}^{k} \frac{k}{\ell} \sum_{(a_1,\ldots,a_\ell)\in\mathcal{A}_{k,\ell}} \mathrm{Tr}\left(H_{a_1}H_{a_2}\cdots H_{a_\ell}\right), \tag{16}$$

and the moment change is $\Delta m_k = \Delta T_k/n$. Hence, instead of computing affected closed walks through full $n \times n$ matrix power, we only need products of $r \times r$ matrices defined over the touched endpoints. Implementation details and complexity analysis are provided in Appendix B.

3.3 Moment-Guided Edge Selection

Given moment changes computed in Sections 3.1 and 3.2, we select edge edits that move the current moment profile toward the target $\mathbf{m}^*$.

Scoring. For a candidate edit ϵ, the expected moment profile after applying this edit is $\mathbf{m} + \Delta\mathbf{m}(\epsilon)$. We score the edit by its normalized distance to the target:

$$\mathrm{score}(\epsilon) = \sum_{k\in\mathcal{K}} \left(\frac{m_k + \Delta m_k(\epsilon) - m_k^*}{\sigma_k} \right)^2, \tag{17}$$

where σ_k is a normalization scale, chosen as either the original moment $m_k(G)$ or the standard deviation of candidate changes $\{\Delta m_k(\epsilon) : \epsilon \in \mathcal{C}_t\}$.

Selection. At each step, we evaluate Eq. 17 over a candidate set $\mathcal{C}_t$ and select

$$\epsilon_t^* = \arg\min_{\epsilon\in\mathcal{C}_t} \mathrm{score}(\epsilon). \tag{18}$$

We apply ϵ_t^*, update $\mathbf{m} \leftarrow \mathbf{m} + \Delta\mathbf{m}(\epsilon_t^*)$, and repeat until the sampling budget is reached. The candidate set can include both edge additions and deletions.

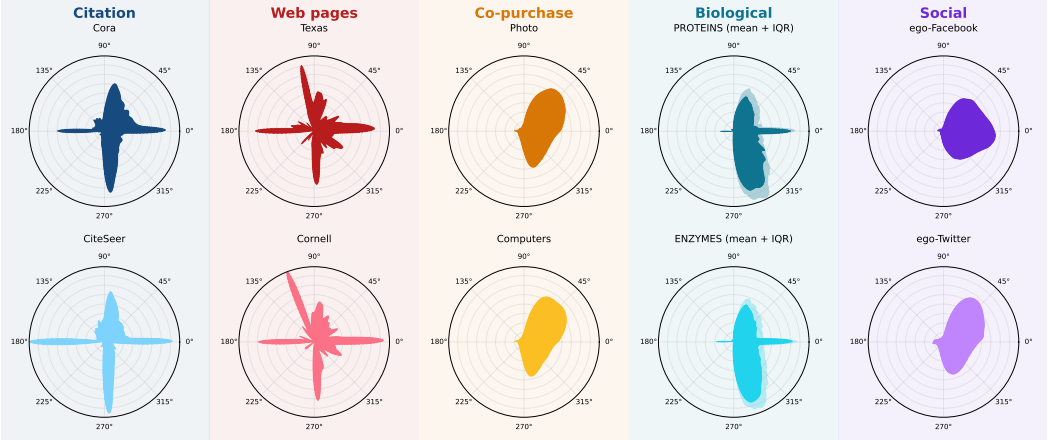


Figure 2: Moment-change fingerprints of real-world graphs. Each profile shows the smoothed edge distribution in the normalized $(\Delta m_2(e), \Delta m_3(e))$ space. Similar profiles indicate similar distributions of local edge-centered motifs. Visualization details are provided in Appendix C.1.

4 MOMENT CHANGES AS STRUCTURAL SIGNATURES

Having established how to compute edge-wise moment changes, we next examine what structural information they encode. For interpretability, we focus on edge deletion. For an existing edge $e = (u, v) \in E$, let $\epsilon_e = (\text{DELETE}, u, v)$ and write $\Delta\mathbf{m}(e) := \Delta\mathbf{m}(\epsilon_e)$ for brevity. We show that these moment changes characterize local edge structure and aggregate into graph-level fingerprints.

Moment changes reveal edge-level structural signatures. The vector $\Delta\mathbf{m}(e)$ describes how deleting e moves the graph in moment space, reflecting the edge's structural role. For visualization, we focus on (m_2, m_3) and normalize each coordinate as detailed in Appendix C.1. Figure 1 connects $\Delta\mathbf{m}(e)$ to graph structure at two levels: representative edge-centered motifs (around the figure examples) illustrate the local meaning of removed edges, while classic graph distributions (colored spills) show how these local patterns aggregate across graphs.

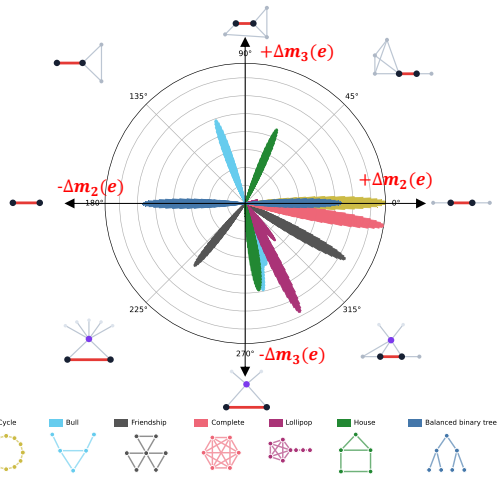


Figure 1: Edge distributions of classic graphs in the $(\Delta m_2, \Delta m_3)$ space. Outer examples show representative edge-centered motifs along different moment-change directions, with the red edge indicating the deleted edge. Inner profiles show the smoothed edge distributions for classic graphs.

Edges within triangles tend to have negative $\Delta m_3(e)$ because their deletion destroys triangles. In contrast, bridge-like edges attached to clustered regions can have positive $\Delta m_3(e)$ because their removal concentrates three-step return probability within the remaining cluster. For $\Delta m_2(e)$, deleting an internal path edge can increase two-step return probabilities, whereas deleting an isolated dyad edge decreases them by isolating both endpoints.

These patterns are also visible in the classic graphs. Graphs with structurally homogeneous edges, such as cycles and complete graphs, concentrate around a narrow range of directions, whereas graphs with multiple edge types, such as the bull graph, exhibit multiple directional modes. Together, these observations show that changes in moments provide interpretable signatures of local edge structures.

Edge-level signatures aggregate into graph-level fingerprints. We next move from individual edges to real-world graphs. As shown in Figure 2, graphs from the same domain often exhibit similar moment-change profiles, while different domains can show distinct patterns. These distributions provide compact graph-level fingerprints of the structural edge types present in a graph.

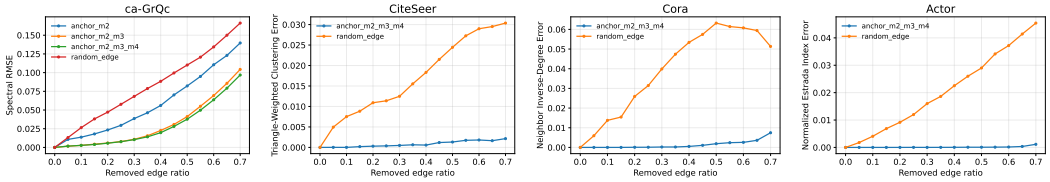


Figure 3: Preservation of four structural properties under moment-preserving edge sampling: *normalized spectrum*, *triangle-weighted clustering*, *neighbor inverse degree*, and *normalized Estrada index*. Curves prefixed with `anchor` denote our method, with m_k indicating the preserved moment orders. Moment-preserving sampling yields smaller deviations than uniform random edge removal across all four properties. For the spectrum, preserving more moment orders further reduces RMSE.

Different fingerprints reveal structural differences across domains. Citation and web-page graphs show broadly similar profiles but differ clearly in the second quadrant. Web-page graphs contain more edges in this region, which are often bridge-like edges connected to clustered regions. This pattern is consistent with hub pages serving navigational roles across otherwise distinct local page clusters, whereas citation edges more often connect papers within the same topical community.

Similar fingerprints suggest shared edge-formation mechanisms. Co-purchase networks and social networks exhibit similar profiles despite belonging to different domains. This similarity may reflect shared-neighborhood mechanisms: products connect through common transactions or categories, while people connect through shared affiliations, interests, or mutual friends (Kossinets & Watts, 2006). Such mechanisms promote dense communities, triangles, and hub–periphery patterns (Leskovec et al., 2009), leading to similar moment-change profiles.

5 Moment-Preserving Sampling Retains Structural Properties

Section 4 shows that edge-wise moment changes encode structural information. We now ask whether preserving the moments also retains structural properties associated with them. Because P normalizes transitions by node degree and $m_k = n^{-1}\operatorname{Tr}(P^k)$ averages over all nodes, *these moments naturally align with normalized or degree-weighted properties rather than raw, size-dependent counts.* This makes moment preservation particularly relevant when reducing graph complexity while retaining structural characteristics, as in graph sparsification (Spielman & Srivastava, 2008), representative graph sampling (Leskovec & Faloutsos, 2006), and graph augmentation (Rong et al., 2019).

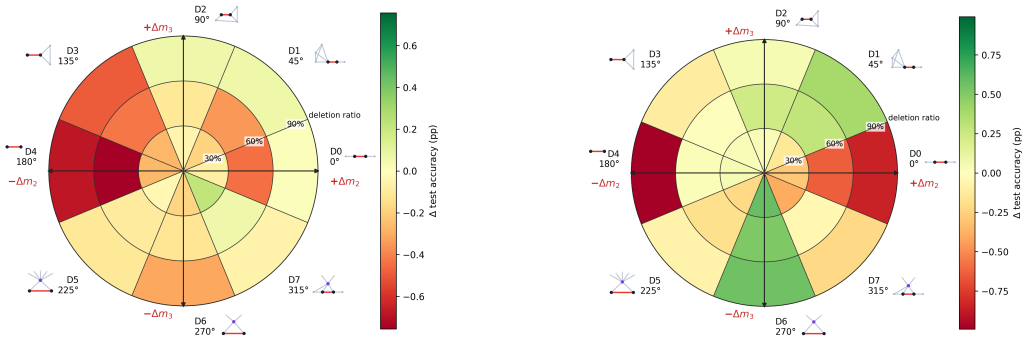
To examine this relationship, we perform *moment-preserving edge sampling*. Let G_0 be the original graph and set the target profile to its moments, $\mathbf{m}^* = \mathbf{m}(G_0)$. We progressively remove edges while minimizing deviation from this target (i.e., ensuring moments stay the same) and track which structural properties remain stable. We study four properties related to the moments and compare their preservation against uniform random edge removal at the same removal ratios. Figure 3 shows representative results, with additional results and derivations are provided in Appendix C.2.

Normalized spectrum. Moments characterize the eigenvalue distribution, and finitely many moments can approximate a graph spectrum (Cohen-Steiner et al., 2018; Kong & Valiant, 2017). Preserving more moments constraints the spectrum more strongly. We measure spectral preservation using the root mean squared error (RMSE) between the eigenvalues of the original and sampled graphs. Figure 3 shows that preserving additional moment orders further reduces the spectral RMSE.

Triangle-weighted clustering coefficient. The third moment m_3 is a degree-weighted sum over triangles and is therefore closely related to clustering. Let $C_i = \frac{2|\Delta_i|}{d_i(d_i-1)}$, $\Delta_i = \big\{\{j,k\} : j,k \in N_i,\ (j,k) \in E\big\}$ denote the local clustering coefficient and the set of triangles incident to node i. This relation motivates the *triangle-weighted* clustering coefficient

$$\widetilde{C}_i = C_i \cdot \mathbb{E}_{\{j,k\}\sim\Delta_i}\left[\frac{1}{\sqrt{d_j d_k}}\right], \tag{19}$$

Compared with standard clustering, $\widetilde{C}_i$ assigns greater weight to triangles involving lower-degree neighbors. We evaluate preservation using the absolute error of its graph-level mean.



(a) Δ Test accuracy on Cora.

(b) Δ Test accuracy on Citeseer.

Figure 4: Effect of removing edges from different moment-change directions on supervised node classification. Edges are divided into eight directions ($D0$–$D7$), each associated with a representative motif. The radius indicates the deletion ratio within each direction, and the heatmap shows the test–accuracy difference from uniform random edge removal under the same budget. Edges connecting training nodes with the same label are excluded from deletion.

Degree structure: accessibility-weighted inverse degree of r-step reachable nodes. The second moment has a direct interpretation in terms of neighboring degrees: $m_2 = \frac{1}{n}\sum_i \mathbb{E}_{j\sim N(i)}[1/d_j]$. Thus, m_2 measures the average inverse degree observed from one-hop neighbors.

More generally, higher even-order moments extend this interpretation to r-step reachable nodes. For an r-step walk, define

$$a_i^{(r)}(j) = d_i(P^r)_{ji} = \sum_{\pi: i \overset{r}{\rightsquigarrow} j} \prod_{v\in \mathrm{Int}(\pi)} \frac{1}{d_v}, \qquad m_{2r} = \frac{1}{n}\sum_i \mathbb{E}\left[\frac{a_i^{(r)}(X_r)}{d_{X_r}}\middle| X_0 = i\right]. \tag{20}$$

Here, $X_0 = i$ and X_r denote the starting and r-step reached nodes, respectively; $\pi : i \overset{r}{\rightsquigarrow} j$ denotes an r-step walk from i to j, and $\mathrm{Int}(\pi)$ its internal nodes. Thus, m_{2r} captures the inverse degree of r-step reachable nodes weighted by their accessibility. In our experiments, we use the $r = 1$ case and measure the absolute error of average neighbor inverse degree.

Diffusion-based connectivity. The moment m_k is the average probability that a k-step random walk returns to its starting node. This return probability reflects how strongly diffusion remains concentrated around its origin: larger values indicate more locally confined diffusion, while smaller values indicate broader spreading through the graph. The collection of moments therefore reflects this diffusion-based connectivity across different walk lengths. To aggregate return behavior across different walk lengths, we use a normalized variant of the Estrada index (Jin & Zafarani, 2020; Estrada, 2002) $\frac{1}{n}EE_P(G) = \frac{1}{n}\sum_{j=1}^n e^{\lambda_j} = \frac{1}{n}\mathrm{Tr}(e^P) = \frac{1}{n}\sum_{k=0}^\infty \frac{\mathrm{Tr}(P^k)}{k!} = \sum_{k=0}^\infty \frac{m_k}{k!}$. Thus, the normalized Estrada index summarizes multi-scale return behavior as a diffusion-based connectivity measure. We measure its preservation using absolute error.

Figure 3 shows that moment-preserving sampling produces smaller deviations than uniform random edge removal across all four properties. Together, these results indicate that controlling spectral moments can retain multiple structural characteristics while the graph is progressively sampled.

6 MOMENT-GUIDED SAMPLING FOR GRAPH LEARNING

Section 4 and 5 show that moments encode and preserve structural information. We next explore the potential of this structural control for graph learning through two case studies on Cora and Citeseer: supervised node classification and graph contrastive learning.

Moment changes reveal structural sensitivity in supervised learning. The moment change $\Delta\mathbf{m}(e)$ characterizes the structural role of an edge, allowing edges with similar local structure to be grouped together. We divide edges into eight moment-change directions, remove different

fractions from each group, and compare against uniform random edge removal under the same budget. As shown in Figure 4, removing many edges from $D4$ consistently degrades performance on both datasets. This degradation is associated with more isolated nodes, which limits neighborhood aggregation in these homophily graphs. These results illustrate how moment changes provide an interpretable way to analyze the sensitivity of supervised learning to different edge structures.

Moment-guided sampling as graph augmentation. Graph contrastive learning relies on augmented views that perturb the graph while retaining useful structural information (Zhu et al., 2021). Since moment-preserving sampling retains several structural properties, we use it to construct an augmented view and compare with GRACE, MVGRL, and GCL-SPAN (Zhu et al., 2020; Hassani & Khasahmadi, 2020; Lin et al., 2022) on unsupervised node classification. As shown in Table 1, our method achieves competitive results on both Citeseer and Cora, despite not being specifically designed for graph contrastive learning. Additional experimental details are provided in Appendix C.3.2.

Table 1: Unsupervised node classification accuracy (%) on Citeseer and Cora. Results are reported as mean ± standard deviation.

Method	Citeseer	Cora
GRACE	71.15 ± 0.73	82.99 ± 1.53
MVGRL	72.41 ± 1.30	84.18 ± 0.98
GCL-SPAN	71.94 ± 0.68	84.07 ± 0.59
Ours	$\mathbf{72.65 \pm 0.65}$	$\mathbf{84.58 \pm 0.56}$

7 Related Work

Graph sampling and sparsification. Graph sampling and sparsification reduce graph complexity while preserving structural or learning-relevant properties (Chen et al., 2023). Existing methods include random edge dropping for regularization and augmentation (Rong et al., 2019), subgraph sampling for scalable training (Zeng et al., 2019), or graph partitioning to retain community structure (Chiang et al., 2019). However, these methods do not explicitly quantify how each edge edit changes a prescribed global structural representation. Our method addresses this gap by measuring edge-wise moment changes and selecting edits that move the graph toward a target moment profile.

Applications of spectral moments. Spectral moments provide compact summaries of graph structure and diffusion. As shown in Appendix A, heat trace and personalized PageRank (PPR) trace can be expressed as weighted sums of moments, connecting moments to multi-scale diffusion behavior. This connection underlies their use in downstream tasks: (1) **graph representation**, where spectral signatures such as the heat trace summarize local and global structure (Tsitsulin et al., 2018), and moments provide interpretable graph embeddings (Jin & Zafarani, 2020); (2) **preservation of graph properties**, where moments approximate graph spectra (Cohen-Steiner et al., 2018) and relate to properties such as spectral radius, clustering, and the Estrada index (Preciado et al., 2013; Jin & Zafarani, 2020; Estrada, 2002); and (3) **graph learning**, where diffusion operators such as PPR and heat kernels, whose traces are weighted sums of moments, improve graph propagation (Gasteiger et al., 2018; 2019). Unlike these uses of moments as graph-level descriptors or diffusion operators, we use their *changes* as edge-level signals for interpretable and controllable graph sampling.

8 Conclusion

We introduce a moment-guided edge sampling framework that connects local edge edits to global graph structure through spectral moments. To compute edge-wise moment changes efficiently, we develop an $O(1)$ combinatorial method for low-order moment and a low-rank method that supports arbitrary orders and batched edits, reducing single-edge computation by a factor of n over direct recomputation for fixed moment orders. These moment changes provide interpretable structural signatures, capturing local motifs and forming graph-level fingerprints. Using them, our framework selects edits that move the graph toward a target moment profile. We further derive connections between moments and structural properties and validate that moment-preserving sampling retains properties such as triangle-weighted clustering. Finally, case studies in supervised node classification and graph contrastive learning suggest that different structural edge types have distinct effects on supervised performance, while moment-guided augmentation yields competitive results in contrastive learning. Overall, spectral moments provide an interpretable and controllable bridge from local edge edits to global graph structure and downstream learning.

AI USE STATEMENT

In this work, we used generative AI tools to formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, design or provide feedback on research methodology and experiments, implement methods, assist with translation, and interpret results. We did not use generative AI tools to generate synthetic datasets, help develop theoretical models or conceptual frameworks, propose or refine hypotheses, clean or reformat datasets, or support qualitative and thematic data analysis. Additionally, we used generative AI tools to create or modify scientific figures or images, create or edit software code, draft and edit parts of the paper, summarize literature, brainstorm ideas, search for information, and improve readability. We reviewed and verified all AI-assisted proofs, claims, and code. We take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

We provide complete derivations, proofs, and pseudocode in Appendix A and B, and detailed experimental settings in Appendix C The anonymized repository is available at `https://anonymous.4open.science/r/Moment-guided-graph-sampling-CEA4`. We package the implementation as a pip-installable Python library with one-command scripts for reproducing all experimental results. It also provides public interfaces covering all components presented in this paper, including combinatorial and low-rank moment-change computation, batched edits, moment-guided sampling, and moment-change fingerprint construction, facilitating direct use and extension in future research. Usage and parameters are documented in the repository's Interface Parameters section. The repository also includes the code required to run all evaluated baselines.

REFERENCES

Nesreen K Ahmed, Jennifer Neville, and Ramana Kompella. Network sampling: From static to streaming graphs. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(2):1–56, 2013.

Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018.

Yuhan Chen, Haojie Ye, Sanketh Vedula, Alex Bronstein, Ronald Dreslinski, Trevor Mudge, and Nishil Talati. Demystifying graph sparsification algorithms in graph properties preservation. *arXiv preprint arXiv:2311.12314*, 2023.

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 & data mining*, pp. 257–266, 2019.

David Cohen-Steiner, Weihao Kong, Christian Sohler, and Gregory Valiant. Approximating the spectrum of a graph. In *Proceedings of the 24th acm sigkdd international conference on knowledge discovery & data mining*, pp. 1263–1271, 2018.

Ernesto Estrada. Characterization of the folding degree of proteins. *Bioinformatics*, 18(5):697–704, 2002.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. *Advances in neural information processing systems*, 32, 2019.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *International conference on machine learning*, pp. 4116–4126. PMLR, 2020.

Shengmin Jin and Reza Zafarani. The spectral zoo of networks: Embedding and visualizing networks with spectral moments. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1426–1434, 2020.

Weihao Kong and Gregory Valiant. Spectrum estimation from samples. 2017.

Gueorgi Kossinets and Duncan J Watts. Empirical analysis of an evolving social network. *science*, 311(5757):88–90, 2006.

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, 2006.

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

Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

Lu Lin, Jinghui Chen, and Hongning Wang. Spectral augmentation for self-supervised learning on graphs. *arXiv preprint arXiv:2210.00643*, 2022.

Lawrence Page. The pagerank citation ranking: Bringing order to the web. technical report. *Stanford Digital Library Technologies Project, 1998*, 1998.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.

Victor M Preciado, Ali Jadbabaie, and George C Verghese. Structural analysis of laplacian spectral properties of large-scale networks. *IEEE Transactions on Automatic Control*, 58(9):2338–2343, 2013.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019.

Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pp. 563–568, 2008.

Anton Tsitsulin, Davide Mottin, Panagiotis Karras, Alexander Bronstein, and Emmanuel Müller. Netlsd: hearing the shape of a graph. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2347–2356, 2018.

Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *stat*, 1050:21, 2018.

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

Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in neural information processing systems*, 33: 5812–5823, 2020.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. *arXiv preprint arXiv:1907.04931*, 2019.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*, 2020.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. In *Proceedings of the web conference 2021*, pp. 2069–2080, 2021.

A Relationships between diffusion kernel and moments

A.1 Heat Trace: Poisson-weighted Moment Generating Function

The normalized heat trace (Tsitsulin et al., 2018) can be seen as the poisson-weighted generating function of moments:

$$\bar{h}(t) = \frac{1}{n}\sum_j e^{-t\mu_j} = \frac{1}{n}\text{tr}(e^{-tL}) = \frac{1}{n}\text{tr}(e^{-t(I-S)}) = e^{-t}\frac{1}{n}\text{tr}(e^{tS}) \tag{21}$$

$$= e^{-t}\sum_{k=0}^{\infty}\frac{t^k}{k!}\frac{1}{n}\text{tr}(S^k) \tag{22}$$

$$= e^{-t}\sum_{k=0}^{\infty}\frac{t^k}{k!}m_k \tag{23}$$

where $L = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, $S = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, which is similar to $P = D^{-1}A$, and share same eigenvalues $\lambda_i \in [-1, 1]$. When $t << 1$, it focus on low-rank, local; when $t >> 1$, it focus on high-rank, global, such as connectivity. Smoother than single moment.

A.2 PPR Trace: Geometric-weighted Moment Generating Function

Since P is a transition matrix with spectral radius $\rho(P) \leq 1$ and $\alpha \in (0, 1)$.By definition of PPR and Neumann series, we have

$$\Pi_\alpha = \alpha(I - (1-\alpha)P)^{-1} = \alpha\sum_{k=0}^{\infty}(1-\alpha)^k P^k, \tag{24}$$

Where Π_α is pagerank value distribution, and α is restart probability.

$$\frac{\text{Tr}(\Pi_\alpha)}{n} = \alpha\sum_{k=0}^{\infty}(1-\alpha)^k m_k. \tag{25}$$

This result demonstrates that moments summarize the global return behavior of diffusion operators, thus preserving moments is related to preserving quantities that underlie PPR diffusion.

B Additional Details for Moment-guided edge sampling

To make our proposed methods in Section 3.1, 3.2 and 3.3 more concrete (e.g., how to update local statistics), we provide more details about the full process in the following sections and pseudocode:

Unified sampling framework	Section B.1, Algorithm 1
Combinatorial change	Section B.2, Algorithm 2
Local-statistic maintenance	Section B.3, Algorithms 3–5
Low-rank change	Section B.4, Algorithms 6–7
Efficient implementation	Section B.6
Complexity and efficiency evaluation	Section B.7

B.1 Unified Sampling Framework

In this paper, we propose two exact moment-delta computation methods: the combinatorial closed-walk delta in Section 3.1 and the algebraic low-rank delta in Section 3.2. They serve different needs. The combinatorial method gives fast closed-form updates for low-order moments, but is difficult to extend to higher-order moments. The algebraic method extends naturally to higher-order moments and batched edits, but has a higher cost. Thus, both methods can be used as interchangeable delta-computation modules in a unified sampling framework.

Algorithm 1 presents this shared framework. Each candidate edge edit is written as $\epsilon = (o, u, v)$, where $o \in \{\text{ADD}, \text{DELETE}\}$ specifies the edit type and (u, v) is the node pair to be edited. The

abstract routine DELTAMOMENT computes the moment delta for a candidate edit, and can be implemented by either COMBINATORIALDELTA in Algorithm 2 or LOWRANKDELTA in Algorithm 6. After an edit ϵ^* is selected, APPLYOPERATION applies it to the graph and updates any state required by the chosen delta method. For COMBINATORIALDELTA, it calls APPLYADD or APPLYDELETE to maintain d_i, W_i, M_{ij}, H_i. For LOWRANKDELTA, no persistent local statistics are required, so APPLYOPERATION simply applies ϵ^* to G.

B.2 COMBINATORIAL CHANGE FOR LOW-ORDER MOMENTS

We present COMBINATORIALDELTA in Algorithm 2. The edge-deletion case is derived in Section 3.1; here we provide the corresponding formulas for edge addition.

Edge addition induced $\Delta m2$. Adding a new edge $(u, v) \notin E$ introduces one new term in S_2, and decreases the weights of existing edges incident to u or v, because their degrees increase. For node u, the degree changes from d_u to $d_u + 1$, so the total change over its existing incident edges is

$$\sum_{k \in \mathcal{N}(u)} \left(\frac{1}{(d_u+1)d_k} - \frac{1}{d_u d_k} \right) = -\frac{W_u}{d_u(d_u+1)}.$$

Combining both endpoints with the new edge contribution gives

$$\Delta S_2^{\text{add}}(u,v) = \underbrace{\frac{1}{(d_u+1)(d_v+1)}}_{\text{new edge}} - \underbrace{\left[\frac{W_u}{d_u(d_u+1)} + \frac{W_v}{d_v(d_v+1)} \right]}_{\text{down-weighted incident edges}}. \tag{26}$$

Edge addition induced $\Delta m3$. Adding $(u, v) \notin E$ creates one new triangle for each common neighbor of u and v. The total mass of these new triangles is $M_{uv}/((d_u+1)(d_v+1))$. At the same time, existing triangles incident to u or v are down-weighted because the endpoint degrees increase. For node u, this change is

$$-\frac{1}{d_u(d_u+1)} \sum_{\{j,k\} \subseteq \mathcal{N}(u), \{j,k\} \in E} \frac{1}{d_j d_k} = -\frac{H_u}{d_u(d_u+1)}.$$

Thus,

$$\Delta S_3^{\text{add}}(u,v) = \underbrace{\frac{M_{uv}}{(d_u+1)(d_v+1)}}_{\text{new triangles}} - \underbrace{\sum_{i \in \{u,v\}} \frac{H_i}{d_i(d_i+1)}}_{\text{down-weighted incident triangles}}. \tag{27}$$

Finally, the normalized moment deltas are

$$\Delta m_2 = \frac{2}{n} \Delta S_2^{\text{add}}(u,v), \qquad \Delta m_3 = \frac{6}{n} \Delta S_3^{\text{add}}(u,v).$$

B.3 MAINTAINING LOCAL STATISTICS

The combinatorial method relies on the local statistics d_i, W_i, M_{ij}, H_i. After each edge edit, these quantities must be updated before evaluating the next candidate, so that the moment deltas remain accurate. Algorithms 5, 3, and 4 give the local update rules for maintaining them without recomputing the whole graph.

DELTAHADD and DELTAHDELETE compute the changes to the triangle masses H_i caused by an edge addition or deletion. APPLYADD and APPLYDELETE then update the graph, degrees, neighborhoods, and the maintained statistics W_i, M_{ij}, H_i. All updates are restricted to the edited endpoints and their neighborhoods.

B.4 LOW-RANK CHANGE FOR GENERAL EDGE EDITS

We represent LOWRANKDELTA in Algorithm 6. In our experiments, we focus on single-edge deletion. For deleting $(u, v) \in E$, assuming $d_u > 1$ and $d_v > 1$, the changed columns are

$$p'_u = \frac{A_{:u} - e_v}{d_u - 1}, \qquad p'_v = \frac{A_{:v} - e_u}{d_v - 1}.$$

Thus,

$$c_u = p'_u - p_u = \frac{p_u - e_v}{d_u - 1}, \qquad c_v = p'_v - p_v = \frac{p_v - e_u}{d_v - 1}.$$

Similarly, for adding a non-edge $(u, v) \notin E$, the changed columns are

$$p'_u = \frac{A_{:u} + e_v}{d_u + 1}, \qquad p'_v = \frac{A_{:v} + e_u}{d_v + 1},$$

and therefore

$$c_u = p'_u - p_u = \frac{e_v - p_u}{d_u + 1}, \qquad c_v = p'_v - p_v = \frac{e_u - p_v}{d_v + 1}.$$

After obtaining the changed columns, we construct C and R and compute H_t. For a batch $\mathcal{B}$, we first apply all edits locally, then set $c_i = P'_{:i} - P_{:i}$ for each touched endpoint $i \in T$. This automatically captures interactions among edited edges through the updated touched columns.

B.5 Interactions and Locality of moment changes

This section characterizes how edge edits interact, and how these interactions control the accuracy of composing single-edge moment changes.

Setting and notation. Let $G = (V, E)$ be a simple, undirected, unweighted graph with a fixed node set of size n. Write $P_G = A_G D_G^{-1}$, where the column corresponding to an isolated node is defined to be zero. Thus P_G is nonnegative and column substochastic, and $\|P_G\|_1 \leq 1$ for the induced matrix 1-norm. No connectivity or minimum-degree constraint is imposed on the resulting graph. For an integer $k \geq 2$ and a present edge e, define

$$m_k(G) = \frac{1}{n} \operatorname{Tr}(P_G^k), \qquad \Delta_k^G(e) = m_k(G - e) - m_k(G). \tag{28}$$

For distinct present edges e, h, their interaction is the symmetric mixed difference

$$\begin{aligned} I_k^G(e, h) &= \Delta_k^{G-h}(e) - \Delta_k^G(e) \\ &= m_k(G - e - h) - m_k(G - e) - m_k(G - h) + m_k(G). \end{aligned} \tag{29}$$

When $I_k^G(e, h) = 0$, it means remove e or h do not change other's moment k change. Let T_e be the endpoint set of e. We use endpoint-set distance,

$$d_G(e, h) = \min_{x \in T_e,\, y \in T_h} d_G(x, y), \tag{30}$$

with distance $+\infty$ between disconnected sets. In particular, incident edges have distance zero. Unless explicitly stated otherwise, all degrees and neighborhoods in a formula are evaluated in its superscript graph G.

B.5.1 Exact locality

Proposition 1 (Locality of interactions)**.** *For every pair of distinct present edges and every integer $k \geq 2$,*

$$2d_G(e, h) > k \quad \Longrightarrow \quad I_k^G(e, h) = 0. \tag{31}$$

Proof. Expand $\operatorname{Tr}(P_G^k)$ as a sum over the length-k closed walks of G. Use the same walks to index the traces for its subgraphs $G - h$, $G - e$, and $G - e - h$, assigning weight zero to a walk that uses a deleted edge. Deleting e only changes transition columns indexed by T_e. Consequently, a walk that avoids T_e has the same weight in G and $G - e$, and also in $G - h$ and $G - e - h$. Its contribution to Eq. 29 cancels. The same argument applies to walks avoiding T_h. A potentially nonzero contribution must therefore visit both endpoint sets. Such a closed walk contains segment from one set to the other and a segment returning to the first set, each of length at least $d_G(e, h)$. Its length is at least $2d_G(e, h)$, proving the claim. This argument also covers deletions that create isolated vertices. □

For a collection of orders $\mathcal{K}$ with $R = \max \mathcal{K}$, all selected moment changes of e are unchanged by deleting h whenever $d_G(e,h) > \lfloor R/2 \rfloor$. For $\mathcal{K} = \{2,3\}$, the locality proposition implies that deleting h leaves the moment changes of e unchanged whenever $d_G(e,h) > 1$. Thus, edge deletions affect other edges' moment changes only locally; beyond this distance, the moment changes remain unchanged and can be reused without recomputation.

The distance condition is sufficient but not necessary for zero interaction. Edges within the locality radius may also have unchanged moment changes. For example, let $G = K_{1,3}$ and choose any two distinct edges e, h. These edges share the center so $d_G(e,h) = 0$. With the node set fixed, each of G, $G-e$, $G-h$, $G-e-h$ consists of a nonempty star and possibly isolated nodes. All four graphs satisfy $m_2 = 1/2$ and $m_3 = 0$, giving $I_2^G(e,h) = I_3^G(e,h) = 0$. Thus, the locality radius identifies a set of potentially affected edges, rather than asserting that every edge within this set is affected.

Corollary 2 (Additivity of separated deletions)**.** *Let $S_1, \dots, S_p \subseteq E(G)$ be pairwise disjoint, and let $R = \max \mathcal{K}$. If $d_G(e,h) > \lfloor R/2 \rfloor$ whenever $e \in S_\ell$, $h \in S_j$, and $\ell \neq j$, then, for every $k \in \mathcal{K}$,*

$$m_k\left(G - \bigcup_{\ell=1}^{p} S_\ell\right) - m_k(G) = \sum_{\ell=1}^{p} \big[m_k(G - S_\ell) - m_k(G)\big]. \tag{32}$$

Proof. By the separation condition, no length-k closed walk can visit deleted-edge endpoints from two distinct sets S_ℓ: such a walk would require length at least $2d_G(e,h) > R \geq k$. Thus, each walk's weight is affected by at most one deletion set, and its weight change is additive across the sets. Summing over all length-k closed walks, with weight zero for walks using deleted edges, and dividing by n proves the claim. □

B.5.2 Interaction bounds for arbitrary moment orders

The locality result identifies when interaction vanish. We now derive an upper bound on $|I_k^G(e,h)|$ for every integer $k \geq 2$, using the notation and conventions introduced above. The argument exploits the fact that deleting an edge changes only the transition columns indexed by its endpoints.

Trace inequality. We first record an inequality that accounts for the columns affected by an edge deletion. Throughout this section, $\|\cdot\|_1$ denotes the induced matrix 1-norm for matrices and the vector 1-norm for column vectors.

Lemma 3 (Trace bound using column support)**.** *Let $B, C \in \mathbf{R}^{n\times n}$. If $C_{:i} = 0$ for every $i \notin T$, then*

$$|\mathrm{Tr}(BC)| \leq \|B\|_1 \sum_{i\in T} \|C_{:i}\|_1. \tag{33}$$

Proof. Let $\mathbf{e}_i$ denote the ith standard basis vector. Since columns outside T are zero,

$$\mathrm{Tr}(BC) = \sum_{i\in T} \mathbf{e}_i^\top BC_{:i}.$$

Consequently,

$$\begin{aligned} |\mathrm{Tr}(BC)| &\leq \sum_{i\in T} \left|\mathbf{e}_i^\top BC_{:i}\right| \\ &\leq \sum_{i\in T} \|BC_{:i}\|_1 \\ &\leq \|B\|_1 \sum_{i\in T} \|C_{:i}\|_1, \end{aligned}$$

which proves the claim. □

Perturbation parameters. For distinct edges $e, h \in E(G)$, define

$$P_{00} = P_G, \qquad P_{10} = P_{G-e}, \qquad P_{01} = P_{G-h}, \qquad P_{11} = P_{G-e-h},$$

and let

$$\begin{aligned} X &= P_{10} - P_{00}, \\ Y &= P_{01} - P_{00}, \\ Z &= P_{11} - P_{10} - P_{01} + P_{00}. \end{aligned} \tag{34}$$

Thus, X and $X + Z$ describe the transition-matrix changes caused by deleting e before and after deleting h, respectively. Similarly, Y and $Y + Z$ describe the changes caused by deleting h before and after deleting e.

Both X and $X + Z$ have nonzero columns only in T_e, whereas Y and $Y + Z$ have nonzero columns only in T_h. It follows that Z has nonzero columns only in $T_e \cap T_h$.

For $i \in T_e$, define

$$\rho_{e,i} = \max\left\{\|X_{:i}\|_1,\ \|(X+Z)_{:i}\|_1\right\},$$

and set

$$s_e = \sum_{i \in T_e} \rho_{e,i}, \qquad \alpha_e = \max_{i \in T_e} \rho_{e,i}. \tag{35}$$

Define $\rho_{h,i}$, s_h, and α_h analogously using Y and $Y + Z$, and let

$$\beta = \sum_{i \in V} \|Z_{:i}\|_1.$$

Here, s_e bounds the total column change associated with deleting e, while α_e bounds its largest column change, accounting for both backgrounds. The parameter β measures the mixed change in the transition matrix itself.

Theorem 4 (Interaction bound for arbitrary moment orders)**.** *For every integer $k \geq 2$,*

$$\boxed{|I_k^G(e,h)| \leq \frac{k\beta + k(k-1)\min\{s_e\alpha_h, s_h\alpha_e\}}{n}}. \tag{36}$$

The bound allows shared endpoints and deletions that create isolated nodes.

Proof. For $s, t \in [0,1]$, define the bilinear interpolation

$$\begin{aligned} M(s,t) &= P_{00} + sX + tY + stZ \\ &= (1-s)(1-t)P_{00} + s(1-t)P_{10} + (1-s)tP_{01} + stP_{11}. \end{aligned} \tag{37}$$

The four coefficients are nonnegative and sum to one. Since each P_{ab} is nonnegative and column substochastic, so is $M(s,t)$. Hence

$$\|M(s,t)\|_1 \leq 1, \qquad \|M(s,t)^j\|_1 \leq 1 \quad \text{for every integer } j \geq 0. \tag{38}$$

The case $j = 0$ uses $M^0 = I$ and $\|I\|_1 = 1$.

Let

$$f(s,t) = \frac{1}{n}\operatorname{Tr}(M(s,t)^k).$$

At the four corners, f equals the corresponding graph moment. Since f is a polynomial in s and t, the fundamental theorem of calculus gives

$$\begin{aligned} I_k^G(e,h) &= f(1,1) - f(1,0) - f(0,1) + f(0,0) \\ &= \int_0^1 \int_0^1 f_{st}(s,t)\, dt\, ds. \end{aligned} \tag{39}$$

We suppress the arguments (s,t) below. The matrix derivatives are

$$M_s = X + tZ, \qquad M_t = Y + sZ, \qquad M_{st} = Z.$$

By the matrix product rule and cyclic invariance of the trace,

$$\begin{aligned} f_s &= \frac{1}{n}\sum_{a=0}^{k-1} \operatorname{Tr}(M^a M_s M^{k-1-a}) \\ &= \frac{k}{n}\operatorname{Tr}(M^{k-1} M_s). \end{aligned}$$

Differentiating once more yields

$$f_{st} = \frac{k}{n}\operatorname{Tr}(M^{k-1}Z) + \frac{k}{n}\sum_{j=0}^{k-2}\operatorname{Tr}(M^j M_t M^{k-2-j} M_s). \tag{40}$$

This identity does not require the matrices to commute.

Since

$$M_s = (1-t)X + t(X+Z),$$

the triangle inequality gives

$$\|(M_s)_{:i}\|_1 \le \rho_{e,i} \qquad (i \in T_e),$$

and all other columns of M_s are zero. Therefore,

$$\sum_{i\in T_e}\|(M_s)_{:i}\|_1 \le s_e, \qquad \|M_s\|_1 \le \alpha_e.$$

Similarly,

$$\sum_{i\in T_h}\|(M_t)_{:i}\|_1 \le s_h, \qquad \|M_t\|_1 \le \alpha_h.$$

Applying Lemma 3 and Eq. 38, we obtain

$$\left|\operatorname{Tr}(M^{k-1}Z)\right| \le \|M^{k-1}\|_1 \sum_i \|Z_{:i}\|_1 \le \beta,$$

$$\left|\operatorname{Tr}(M^j M_t M^{k-2-j} M_s)\right| \le \|M^j M_t M^{k-2-j}\|_1 \sum_{i\in T_e}\|(M_s)_{:i}\|_1 \le \alpha_h s_e.$$

Alternatively, cyclically rotating the trace gives

$$\operatorname{Tr}(M^j M_t M^{k-2-j} M_s) = \operatorname{Tr}(M^{k-2-j} M_s M^j M_t),$$

which yields the bound $\alpha_e s_h$. Thus, each summand is bounded in absolute value by $\min\{s_e\alpha_h, s_h\alpha_e\}$.

There are $k-1$ summands in Eq. 40, so

$$|f_{st}(s,t)| \le \frac{k\beta + k(k-1)\min\{s_e\alpha_h, s_h\alpha_e\}}{n}$$

throughout $[0,1]^2$. Taking absolute values in Eq. 39 and integrating this uniform bound proves the result.

□

Explicit degree parameters. The parameters above can be computed from endpoint degrees. Deleting an incident edge at a node of degree $d \ge 1$ changes its transition column by an amount whose ℓ_1-norm is

$$q(d) = \begin{cases} 1, & d = 1, \\ 2/d, & d \ge 2. \end{cases} \tag{41}$$

Indeed, when $d \ge 2$, the deleted neighbor loses probability $1/d$, while the remaining $d-1$ neighbors each gain $1/[d(d-1)]$. The total absolute change is therefore $2/d$. When $d = 1$, the column changes from a unit vector to zero, giving norm one.

If e and h have disjoint endpoint sets, then $Z = 0$ and $\beta = 0$. Writing $d_i = d_G(i)$, we have

$$s_e = \sum_{i\in T_e} q(d_i), \qquad \alpha_e = \max_{i\in T_e} q(d_i),$$

and analogously for h. Let

$$d_e = \min_{i\in T_e} d_i, \qquad d_h = \min_{i\in T_h} d_i.$$

Since $q(d) \le 2/d$ for all $d \ge 1$,

$$s_e \le \frac{4}{d_e}, \quad \alpha_e \le \frac{2}{d_e}, \quad s_h \le \frac{4}{d_h}, \quad \alpha_h \le \frac{2}{d_h}.$$

Theorem 4 therefore implies

$$\boxed{|I_k^G(e,h)| \le \frac{8k(k-1)}{n\, d_e d_h}.} \tag{42}$$

This bound also covers degree-one endpoints.

If $e = uv$ and $h = uw$ share an endpoint u, then $d_u \ge 2$ because both edges are present. Deleting e from G changes column u by ℓ_1-norm $q(d_u)$, whereas deleting e from $G - h$ changes that column by ℓ_1-norm $q(d_u - 1)$. The same two values arise when the roles of e and h are exchanged. Since $q(1) = q(2) = 1$ and $q(D) = 2/D$ for $D \ge 2$, we have

$$q(d_u - 1) \ge q(d_u), \qquad q(d_u - 1) = \begin{cases} 1, & d_u = 2, \\ \dfrac{2}{d_u - 1}, & d_u \ge 3. \end{cases}$$

Thus, the maximum column-change norm at the shared endpoint is exactly $q(d_u - 1)$. The exact parameter values are therefore

$$\begin{aligned} s_e &= q(d_u - 1) + q(d_v), & \alpha_e &= \max\{q(d_u - 1), q(d_v)\}, \\ s_h &= q(d_u - 1) + q(d_w), & \alpha_h &= \max\{q(d_u - 1), q(d_w)\}, \\ \beta &= \begin{cases} 1, & d_u = 2, \\ \dfrac{4}{d_u(d_u - 1)}, & d_u \ge 3. \end{cases} \end{aligned} \tag{43}$$

The expressions for s_e and α_e follow because deleting h reduces the degree of u by one and leaves the transition column at v unchanged. Similarly, deleting e reduces the degree of u by one and leaves the transition column at w unchanged, giving the expressions for s_h and α_h. To verify the expression for β, recall that

$$Z = P_{G-e-h} - P_{G-e} - P_{G-h} + P_G, \qquad \beta = \sum_i \|Z_{:i}\|_1.$$

The matrix Z is supported only on columns in $T_e \cap T_h = \{u\}$, so $\beta = \|Z_{:u}\|_1$. For $d_u \ge 3$, its entries at rows v and w are

$$Z_{vu} = Z_{wu} = -\frac{1}{d_u - 1} + \frac{1}{d_u} = -\frac{1}{d_u(d_u - 1)}.$$

At each remaining neighbor $x \in N_G(u) \setminus \{v, w\}$, the entry is

$$Z_{xu} = \frac{1}{d_u - 2} - \frac{2}{d_u - 1} + \frac{1}{d_u} = \frac{2}{d_u(d_u - 1)(d_u - 2)}.$$

All other entries are zero. Since there are $d_u - 2$ remaining neighbors, summing the absolute values gives

$$\begin{aligned} \beta &= \frac{2}{d_u(d_u - 1)} + (d_u - 2)\frac{2}{d_u(d_u - 1)(d_u - 2)} \\ &= \frac{4}{d_u(d_u - 1)}. \end{aligned}$$

If $d_u = 2$, deleting both edges isolates u. Under the zero-column convention for isolated nodes, the four relevant columns are

$$(P_G)_{:u} = \frac{1}{2}(\mathbf{e}_v + \mathbf{e}_w), \qquad (P_{G-e})_{:u} = \mathbf{e}_w, \qquad (P_{G-h})_{:u} = \mathbf{e}_v, \qquad (P_{G-e-h})_{:u} = 0.$$

Consequently,

$$Z_{:u} = 0 - \mathbf{e}_w - \mathbf{e}_v + \frac{1}{2}(\mathbf{e}_v + \mathbf{e}_w) = -\frac{1}{2}(\mathbf{e}_v + \mathbf{e}_w),$$

and hence $\beta = \|Z_{:u}\|_1 = 1$.

Corollary 5 (A uniform degree-based bound)**.** *Suppose that every node in $T_e \cup T_h$ has degree at least $d \geq 2$ in G. Then, for every integer $k \geq 2$,*

$$\boxed{|I_k^G(e,h)| \leq \frac{1}{n}\left[\frac{4k}{d(d-1)} + \frac{8k(k-1)}{(d-1)^2}\right].} \tag{44}$$

If $T_e \cap T_h = \varnothing$, the sharper bound

$$|I_k^G(e,h)| \leq \frac{8k(k-1)}{nd^2}$$

holds.

Proof. Recall that $q(D) \leq 2/D$ for every integer $D \geq 1$. If the edges share an endpoint u, then $d_u - 1 \geq d - 1 \geq 1$, so

$$q(d_u - 1) \leq \frac{2}{d_u - 1} \leq \frac{2}{d-1}.$$

The other endpoint terms satisfy

$$q(d_v), q(d_w) \leq \frac{2}{d} \leq \frac{2}{d-1}.$$

Substituting these inequalities into Eq. 43 gives

$$s_e, s_h \leq \frac{4}{d-1}, \qquad \alpha_e, \alpha_h \leq \frac{2}{d-1}.$$

If $d_u \geq 3$, then

$$\beta = \frac{4}{d_u(d_u-1)} \leq \frac{4}{d(d-1)}.$$

If $d_u = 2$, the condition $2 \leq d \leq d_u$ forces $d = 2$, and

$$\beta = 1 \leq \frac{4}{d(d-1)}.$$

Thus, the required bound on β also holds when deleting both edges isolates the shared endpoint.

If the edges have disjoint endpoint sets, then $\beta = 0$. Each endpoint has degree at least d, and the disjoint-endpoint parameter formulas give

$$s_e, s_h \leq \frac{4}{d} \leq \frac{4}{d-1}, \qquad \alpha_e, \alpha_h \leq \frac{2}{d} \leq \frac{2}{d-1}.$$

Therefore, for either endpoint configuration,

$$s_e, s_h \leq \frac{4}{d-1}, \qquad \alpha_e, \alpha_h \leq \frac{2}{d-1}, \qquad \beta \leq \frac{4}{d(d-1)}.$$

In particular,

$$\min\{s_e\alpha_h, s_h\alpha_e\} \leq \frac{8}{(d-1)^2}.$$

Applying Theorem 4 yields

$$\begin{aligned} |I_k^G(e,h)| &\leq \frac{k\beta + k(k-1)\min\{s_e\alpha_h, s_h\alpha_e\}}{n} \\ &\leq \frac{1}{n}\left[\frac{4k}{d(d-1)} + \frac{8k(k-1)}{(d-1)^2}\right], \end{aligned}$$

which proves Eq. 44. For disjoint endpoint sets, the stronger estimates $\beta = 0$, $s_e, s_h \leq 4/d$, and $\alpha_e, \alpha_h \leq 2/d$ instead give

$$|I_k^G(e,h)| \leq \frac{8k(k-1)}{nd^2}.$$

□

B.6 Efficient Implementation

Moment-guided sampling has two main computational bottlenecks: *evaluating moment changes for individual candidate edits* and *repeatedly rescoring candidates as the graph evolves*. We address the first by exploiting the endpoint structure of the low-rank formulation, computing only the transitions needed between touched endpoints in Section B.6.1. For the second, the locality and bounded interactions in Section B.5 show that an edge edit affects other candidates primarily within a local region. Motivated by this observation, we introduce a lazy top-k rescoring strategy (Section B.6.2) for both the combinatorial and low-rank methods, reducing the number of exact candidate evaluations while empirically preserving sampling quality (Section B.7).

B.6.1 Endpoint low-rank computation

In practice, we often use single-edge deletion, i.e., $|\mathcal{B}| = 1$, to avoid interactions among multiple simultaneous edits. For deleting an edge (u, v), the touched endpoint set is $T = \{u, v\}$ so $r = 2$. Then each $H_t = R^\top P^t C$ is only a 2×2 matrix. However, sparse propagation first computes $P^t C \in \mathbb{R}^{n\times 2}$ over all n nodes and then keeps only the two rows indexed by u and v. This costs $O(Km)$ and is wasteful when only four entries of each H_t are needed. We therefore compute these entries directly from endpoint-to-endpoint transition probabilities.

For $a \in \{u, v\}$, one entry of H_t is

$$H_t(a, u) = e_a^\top P^t c_u \tag{45}$$

$$= e_a^\top P^t \frac{Pe_u - e_v}{d_u - 1} \tag{46}$$

$$= \frac{e_a^\top P^{t+1} e_u - e_a^\top P^t e_v}{d_u - 1}. \tag{47}$$

Define $s_t(a, b) = e_a^\top P^t e_b$. Then

$$H_t(a, u) = \frac{s_{t+1}(a, u) - s_t(a, v)}{d_u - 1}. \tag{48}$$

Similarly, the full matrix can be written as

$$H_t = \begin{bmatrix} \dfrac{s_{t+1}(u,u) - s_t(u,v)}{d_u - 1} & \dfrac{s_{t+1}(u,v) - s_t(u,u)}{d_v - 1} \\ \dfrac{s_{t+1}(v,u) - s_t(v,v)}{d_u - 1} & \dfrac{s_{t+1}(v,v) - s_t(v,u)}{d_v - 1} \end{bmatrix}. \tag{49}$$

If deleting (u, v) isolates an endpoint, e.g., $d_u = 1$, we use the direct column change $c_u = -e_v$, so $H_t(a, u) = -s_t(a, v)$.

Thus, computing H_t reduces to estimating endpoint transition probabilities $s_t(a, b)$ instead of propagating the full sparse product $P^t C$ over the whole graph. This only requires propagation within the local neighborhoods of the endpoints. The cost becomes

$$O\left(\sum_{t=0}^{K} \mathrm{vol}\left(B_t(u) \cup B_t(v)\right)\right),$$

where $B_t(u)$ is the t-hop neighborhood of u, and $\mathrm{vol}(\cdot)$ denotes the sum of degrees in the set. For low-order moments, this local propagation remains small: to compute m_k, we only need transition probabilities up to order $t = k - 1$.

B.6.2 Lazy Top-K candidate rescoring

Exhaustively rescoring all candidates after every edge edit is expensive, especially when the sampling budget grows with graph size. Motivated by the locality and bounded interactions in Section B.5, we use a lazy top-K strategy that avoids repeatedly evaluating all candidates. We first compute the scores of all candidate edges on the original graph and store them in a min-heap. At each sampling step, we retrieve the K candidates with the smallest cached scores and recompute

only their scores on the current graph. We then select the candidate with the smallest updated score, apply the edit, and update its entry in the heap, while the remaining candidates retain their cached scores. The selected edge is removed from the heap, and the procedure repeats until the sampling budget is reached. This reduces each step from rescoring all m candidates to evaluating only $K \ll m$ candidates, trading exact greedy selection for substantially lower computation. We evaluate the resulting sampling quality and runtime in Section B.7.

B.7 Complexity and efficiency evaluation

We evaluate the efficiency of moment-guided sampling from both theoretical and empirical perspectives. Section B.7.1 shows that the combinatorial method evaluates low-order moment changes in $O(1)$ time per candidate, while the low-rank method reduces single-edge computation from $O(Kmn)$ to $O(Km)$. Section B.7.2 shows that the heuristic lazy top-K strategy introduced in Section B.6.2 retains comparable property preservation. Finally, Section B.7.3 shows that lazy rescoring substantially reduces runtime, achieving over $2\times$ speedup in most settings and up to $6.45\times$.

B.7.1 Complexity analysis

We compare the cost of computing moment changes up to order K. A direct method in Eq. 1 recomputes the matrix powers of the updated transition matrix after each edge removal. This costs $O(Kmn)$ time if sparse matrix multiplication is used and the powers become dense, and $O(Kn^3)$ time in the dense worst case. It also requires $O(n^2)$ space. For the combinatorial method, once the local statistics d_i, W_i, M_{ij}, H_i are maintained, computing Δm_2 and Δm_3 for one candidate edge only requires constant-time lookups and arithmetic, i.e., $O(1)$ time per candidate. Its space cost is $O(n + |\mathcal{P}|)$, where $\mathcal{P} = \{(i, j) : \mathcal{N}(i) \cap \mathcal{N}(j) \neq \emptyset\}$ stores pairs with nonzero weighted common-neighbor mass; in the worst case, this becomes $O(n^2)$. For the low-rank method, let T be the touched endpoint set and $r = |T|$. Constructing the low-rank update C only depends on the old and new neighbors of touched endpoints, costing $\tau = O(\sum_{i\in T}(d_i + d'_i))$. Computing all small matrices $H_t = R^\top P^t C$ for $t = 0, \ldots, K-1$ by sparse propagation costs $O(Kmr)$. Eq. 16 contains $O(2^K)$ trace-product terms, and evaluating each product requires at most $O(Kr^3)$ time, giving $O(K2^K r^3)$ time for the trace expansion. For fixed moment orders, the terms and coefficients can be precomputed once, although the matrix product must still be evaluated for each candidate edit. Thus, the total time complexity is $O(\tau + Kmr + K2^K r^3)$, with space complexity $O(m + \tau + Kr^2)$. For fixed K and small r, the cost is dominated by the $O(Kmr)$ sparse propagation.

For a single-edge removal $e = (u, v)$, we have $r = 2$ and $\tau = O(d_u + d_v)$. Per candidate, direct power computation costs $O(Kmn)$ with sparse propagation, or $O(Kn^3)$ in the dense worst case. In contrast, the combinatorial method costs $O(1)$, and the low-rank method costs $O(Km + K2^K)$. For fixed moment orders, the $O(K2^K)$ term is constant with respect to graph size, reducing the low-rank cost to $O(Km)$, or $O(m)$ for fixed K. The corresponding space costs are $O(n^2)$, $O(n + |\mathcal{P}|)$, and $O(m + d_u + d_v + K)$, respectively. Thus, the combinatorial method is the most efficient for m_2 and m_3, while the low-rank method is more general and remains substantially cheaper than direct recomputation. For fixed K, its cost is reduced by a factor of n relative to direct computation. The low-rank computation for single-edge removal can be further accelerated using the endpoint-local implementation in Appendix B.6.

B.7.2 Lazy top-K preserves structural properties

To evaluate sampling quality of lazy top-K rescoring, we examine the same structural properties as in Section 5. Figure 5 compares original moment-guided sampling with lazy top-K variants using different values of K. Larger K more closely matches original sampling, and $K = 25$ already achieves nearly identical property preservation.

B.7.3 Runtime comparison

We compare the preprocessing and per-deletion runtime of direct trace, combinatorial, low-rank, and hybrid moment-change computation, together with their lazy top-K variants. Table 2 reports the results on Cora using a single CPU thread. The results show four main trends:

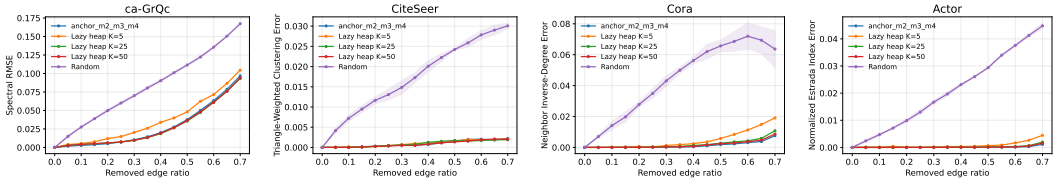


Figure 5: Structural-property preservation under original full-rescoring method and lazy top-K candidate rescoring. The `anchor_m2_m3_m4` curve denotes the original method, using the same moment-preserving setting as in Figure 3. Each lazy variant preserves the same moment orders but rescoring only the top-K candidates at each step. Lazy top-K closely tracks original method, with larger K yielding closer agreement; $K = 25$ already nearly matches the original baseline. Random edge removal produces substantially larger deviations.

Table 2: Sampling efficiency on Cora using a single CPU thread. *Direct trace* computes edge-wise moment changes by directly evaluating the trace difference in Eq. 1; due to its high cost, it is measured over two sequential deletions, while the other methods are evaluated over 100 deletion. Per-deletion time excludes preprocessing. *Hybrid* uses the combinatorial method for m_2, m_3 and the low-rank method for m_4. A dash in the lazy top-K column denotes the original full-rescoring method, while a numeric K denotes lazy top-K rescoring. Speedup is measured relative to the corresponding full-rescoring configuration using the same backend and moment orders.

Moment orders	Backend	Lazy top-K	Time cost (s)		Speedup
			Preprocessing	Per deletion	
(m_2, m_3)	Direct trace	–	0.197	50.16513	–
	Combinatory	–	0.100	0.00067	–
		5	0.101	0.00028	2.37×
		25	0.098	0.00036	1.86×
		50	0.100	0.00041	1.62×
	Low-rank	–	0.098	0.03192	–
		5	0.133	0.00728	4.38×
		25	0.133	0.00760	4.20×
		50	0.132	0.00793	4.02×
(m_2, m_3, m_4)	Direct trace	–	0.104	220.91917	–
	Hybrid	–	0.107	0.04942	–
		5	0.245	0.00767	6.45×
		25	0.170	0.00832	5.94×
		50	0.165	0.00911	5.42×
	Low-rank	–	0.101	0.04895	–
		5	0.154	0.00759	6.45×
		25	0.153	0.00813	6.02×
		50	0.154	0.00889	5.51×

1. Both the combinatorial and low-rank methods are substantially faster than direct trace re-computation.
2. For low-order moments, the combinatorial method is much faster than low-rank method. For (m_2, m_3), it is nearly $50\times$ faster because maintained local statistics enable constant-time candidate evaluation.
3. The hybrid method does not improve runtime over pure low-rank computation for (m_2, m_3, m_4) and is slightly slower in our measurements. This is because the low-rank cost is dominated by the highest moment order: m_4 already requires the necessary H_t matrices, while obtaining lower-order moments from them adds little additional cost.
4. Lazy top-K rescoring further reduces runtime, achieving $1.62\times$–$6.45\times$ speedup over full rescoring, with smaller K consistently yielding greater speedup.

C EXPERIMENTAL DETAILS

This section provides additional experimental details. Appendix C.1 describes the construction of the visualizations in Figures 1 and 2. Appendix C.2 derives the relationships between spectral moments and the graph properties introduced in Section 5. Appendix C.3 provides the experimental settings for the graph learning tasks.

C.1 EDGE MOMENT DIRECTION PROFILES

Edge-deletion moment vectors. Let $\mathcal{K} = \{k_1, \ldots, k_p\}$ be an ordered collection of $p \geq 2$ selected moment orders, defining the moment representation

$$\mathbf{m}_{\mathcal{K}}(G) = \begin{bmatrix} m_{k_1}(G) \\ \vdots \\ m_{k_p}(G) \end{bmatrix} \in \mathbb{R}^p. \tag{50}$$

For each edge $e \in E$, let $G^{-e} = (V, E \setminus \{e\})$ denote the graph obtained by deleting only e. The deletion-induced change in the k-th moment is

$$\Delta m_k(e) = m_k(G^{-e}) - m_k(G). \tag{51}$$

All edges are evaluated independently relative to the same original graph G, providing a common reference for their structural effects. The *edge-deletion moment vector* is

$$\boldsymbol{\Delta}_e = \mathbf{m}_{\mathcal{K}}(G^{-e}) - \mathbf{m}_{\mathcal{K}}(G) = \begin{bmatrix} \Delta m_{k_1}(e) \\ \vdots \\ \Delta m_{k_p}(e) \end{bmatrix}. \tag{52}$$

These moment changes can be computed using the combinatorial formulas for the supported orders (Section 3.1) or the algebraic method for arbitrary orders (Section 3.2).

Normalization. Because the selected moments may have different numerical scales, we normalize each coordinate by the corresponding moment of the original graph. For this relative normalization, we assume $m_k(G) > 0$ for every $k \in \mathcal{K}$. The *normalized edge-deletion moment vector* is

$$\widetilde{\boldsymbol{\Delta}}_e = \begin{bmatrix} \Delta m_{k_1}(e)/m_{k_1}(G) \\ \vdots \\ \Delta m_{k_p}(e)/m_{k_p}(G) \end{bmatrix}. \tag{53}$$

This normalized expresses each coordinate as a relative change from the original graph and prevents either moment from dominating solely because of its numerical scale.

Moment direction and magnitude. We decompose each nonzero normalized vector into its magnitude and unit direction. Its magnitude is

$$r_e = \left\| \widetilde{\boldsymbol{\Delta}}_e \right\|_2 = \sqrt{\sum_{k \in \mathcal{K}} \left(\frac{\Delta m_k(e)}{m_k(G)} \right)^2}, \tag{54}$$

and, for $r_e > 0$, its direction is

$$\mathbf{u}_e = \frac{\widetilde{\boldsymbol{\Delta}}_e}{r_e} \in \mathbb{S}^{p-1}, \qquad \mathbb{S}^{p-1} = \{\mathbf{u} \in \mathbb{R}^p : \|\mathbf{u}\|_2 = 1\}. \tag{55}$$

The magnitude r_e measures the strength of structural change induced by deleting edge e, whereas $\mathbf{u_e}$ describes the relative direction of that change. The direction space is the unit circle for $p = 2$ and the unit sphere or hypersphere for higher dimensions. Zero vectors have no defined direction, so the directional profile uses the edge set $E_+ = \{e \in E : r_e > 0\}$. The following density normalization assumes $E_+ \neq \varnothing$.

Directional density estimation. We measure the separation between two unit directions using their spherical geodesic distance,

$$d_{\mathbb{S}}(\mathbf{u}, \mathbf{v}) = \arccos(\mathbf{u}^\top \mathbf{v}) \in [0, \pi]. \tag{56}$$

To reduce sensitivity to finite-sample noise and discretization boundaries, we smooth the edge directions using a Gaussian kernel based on this distance,

$$K_h(\mathbf{u}, \mathbf{u}_e) = \exp\left[-\frac{1}{2}\left(\frac{d_{\mathbb{S}}(\mathbf{u}, \mathbf{u}_e)}{h}\right)^2\right], \tag{57}$$

where h is the angular bandwidth. We use $h = 3^\circ = \pi/60$ in the reported experiments.

For $p = 2$, write $\mathbf{u}(\theta) = (\cos\theta, \sin\theta)^\top$. Then

$$d_{\mathbb{S}}(\mathbf{u}(\theta), \mathbf{u}(\phi)) = \left|\left((\theta - \phi + \pi) \bmod 2\pi\right) - \pi\right|. \tag{58}$$

Since the kernel uses the squared distance, this definition reduces exactly to the circular Gaussian kernel in two dimensions.

Assigning equal weight to every edge in E_+, we define the unnormalized directional profile as

$$q_G(\mathbf{u}) = \sum_{e \in E_+} K_h(\mathbf{u}, \mathbf{u}_e). \tag{59}$$

Its normalized continuous density with respect to spherical surface measure dS is

$$p_G(\mathbf{u}) = \frac{q_G(\mathbf{u})}{\displaystyle\int_{\mathbb{S}^{p-1}} q_G(\mathbf{v})\, dS(\mathbf{v})}, \qquad \int_{\mathbb{S}^{p-1}} p_G(\mathbf{u})\, dS(\mathbf{u}) = 1. \tag{60}$$

Thus, p_G characterizes the distribution of nonzero edge moment directions in G. For $p = 2$, the surface measure becomes $d\theta$, recovering normalization over the unit circle.

Directional discretization. For numerical evaluation, partition $\mathbb{S}^{p-1}$ into cells $\{C_j\}_{j=0}^{J-1}$, with representative directions $\mathbf{u}_j$ and surface measures $a_j = \int_{C_j} dS > 0$. The normalization integral can be approximated by

$$\int_{\mathbb{S}^{p-1}} q_G(\mathbf{u})\, dS(\mathbf{u}) \approx \sum_{j=0}^{J-1} a_j q_G(\mathbf{u}_j). \tag{61}$$

We define the normalized discrete weight of cell j as

$$w_{G,j} = \frac{a_j q_G(\mathbf{u}_j)}{\displaystyle\sum_{\ell=0}^{J-1} a_\ell q_G(\mathbf{u}_\ell)}, \qquad \sum_{j=0}^{J-1} w_{G,j} = 1. \tag{62}$$

For sufficiently fine cells, this weight approximates the probability mass of the corresponding directional region,

$$w_{G,j} \approx p_G(\mathbf{u}_j) a_j \approx \int_{C_j} p_G(\mathbf{u})\, dS(\mathbf{u}). \tag{63}$$

For equal-area cells, the common factor a_j cancels from the discrete normalization.

Area-normalized visualization. For visualization, we specialize to $\mathcal{K} = \{2, 3\}$ and represent the direction space by the unit circle. Using $\mathbf{u}(\theta) = (\cos\theta, \sin\theta)^\top$, we abbreviate $q_G(\theta) = q_G(\mathbf{u}(\theta))$ and $p_G(\theta) = p_G(\mathbf{u}(\theta))$. We evaluate the profile on a uniform angular grid

$$\Theta = \{\theta_j = j\delta_\theta\}_{j=0}^{J-1}, \qquad \delta_\theta = \frac{2\pi}{J}. \tag{64}$$

We set $\delta_\theta = 0.5^\circ = \pi/360$, giving $J = 720$ sectors. Each sector has angular measure $a_j = \delta_\theta$, so the general discrete weights reduce to

$$w_{G,j} = \frac{q_G(\theta_j)}{\sum_{\ell=0}^{J-1} q_G(\theta_\ell)}, \qquad \sum_{j=0}^{J-1} w_{G,j} = 1. \tag{65}$$

The corresponding quadrature approximation is

$$\int_0^{2\pi} q_G(\theta)\, d\theta \approx \delta_\theta \sum_{\ell=0}^{J-1} q_G(\theta_\ell). \tag{66}$$

For a sufficiently fine grid, the sector weight satisfies

$$w_{G,j} \approx p_G(\theta_j)\delta_\theta \approx \int_{\theta_j-\delta_\theta/2}^{\theta_j+\delta_\theta/2} p_G(\theta)\, d\theta, \tag{67}$$

where the density is interpreted periodically at the endpoints of the angular interval. We encode each sector's probability mass by its area, choosing the radius $R_{G,j}$ to satisfy

$$\frac{1}{2} R_{G,j}^2 \delta_\theta = w_{G,j}. \tag{68}$$

This gives

$$R_{G,j} = \sqrt{\frac{2w_{G,j}}{\delta_\theta}}. \tag{69}$$

The total visualization area is therefore

$$\frac{1}{2} \sum_{j=0}^{J-1} R_{G,j}^2 \delta_\theta = \sum_{j=0}^{J-1} w_{G,j} = 1. \tag{70}$$

Hence, all graphs have equal visualization area, and differences in shape reflect their directional distributions rather than their number of edges.

Graph collections. For a collection $\mathcal{G} = \{G_1, \ldots, G_N\}$, such as PROTEINS or ENZYMES, we compute the normalized cell masses $\{w_{G_g,j}\}_{j=0}^{J-1}$ separately for each graph. Consequently, every graph contributes equally to the collection-level mean, regardless of its size. The mean profile is

$$\overline{w}_{\mathcal{G},j} = \frac{1}{N} \sum_{g=1}^{N} w_{G_g,j}. \tag{71}$$

Variation across graphs is summarized by the pointwise interquartile interval

$$\left[Q_{0.25}\left(\{w_{G_g,j}\}_{g=1}^N\right), Q_{0.75}\left(\{w_{G_g,j}\}_{g=1}^N\right) \right]. \tag{72}$$

For the two-dimensional visualization, the solid curve is obtained by applying Eq. 69 to the mean masses $\overline{w}_{\mathcal{G},j}$. The shaded band is obtained by applying the same radius transformation to the lower and upper pointwise quartiles.

Classic graphs. We provide the specifications of the classic graphs used in Figure 1.

- Complete graph: 20 nodes;
- Cycle: 60 nodes;
- Path: 60 nodes;
- Lollipop graph: 30 nodes with a clique of size 15;
- House graph: 5 nodes;
- Bull graph: 5 nodes;
- Friendship graph: 41 nodes with 20 blades, consisting of one center node and 20 triangles;
- Balanced binary tree: 31 nodes, depth 4, and 30 edges;

C.2 Derivation and evaluation of structural properties

This section derives and evaluates the relationships between spectral moments and the graph properties introduced in Section 5.

C.2.1 Derivations of structural properties

Triangle-weighted clustering coefficient. For an undirected graph $G = (V, E)$, let N_i denote the neighborhood of node i, $d_i = |N_i|$, and

$$\Delta_i = \big\{\{j, k\} : j, k \in N_i,\ (j, k) \in E\big\}$$

be the set of triangles incident to i, represented by the edges among its neighbors. The standard local clustering coefficient is

$$C_i = \frac{2|\Delta_i|}{d_i(d_i - 1)},$$

with $C_i = 0$ when $d_i < 2$.

The third spectral moment of the random-walk matrix $P = D^{-1}A$ can be written as

$$\begin{aligned} m_3 &= \frac{1}{n}\operatorname{Tr}(P^3) = \frac{1}{n}\sum_i (P^3)_{ii} \\ &= \frac{1}{n}\sum_i \sum_{j,k} \frac{A_{ij}A_{jk}A_{ki}}{d_i d_j d_k} \\ &= \frac{1}{n}\sum_i \frac{2}{d_i} \sum_{\{j,k\}\in\Delta_i} \frac{1}{d_j d_k} \\ &= \frac{1}{n}\sum_i \frac{2|\Delta_i|}{d_i} \mathbb{E}_{\{j,k\}\sim\Delta_i}\left[\frac{1}{d_j d_k}\right] \\ &= \frac{1}{n}\sum_i C_i(d_i - 1)\mathbb{E}_{\{j,k\}\sim\Delta_i}\left[\frac{1}{d_j d_k}\right]. \end{aligned} \tag{73}$$

When the degrees within each local triangle are comparable, i.e., $d_i \approx d_j \approx d_k$ for $\{j, k\} \in \Delta_i$, and the degrees are not too small, we have

$$C_i(d_i - 1)\mathbb{E}_{\{j,k\}\sim\Delta_i}\left[\frac{1}{d_j d_k}\right] \approx C_i \mathbb{E}_{\{j,k\}\sim\Delta_i}\left[\frac{1}{\sqrt{d_j d_k}}\right] = \widetilde{C}_i. \tag{74}$$

Thus, the proposed score can be viewed as a moment-aligned weighted local clustering statistic that assigns larger weights to triangles involving low-degree neighbors.

We use mean absolute error to measure:

$$E_{\text{clust}}(G', G) = \left| \frac{1}{|V'|}\sum_{i\in V'} \widetilde{C}_i(G') - \frac{1}{|V|}\sum_{i\in V} \widetilde{C}_i(G) \right|. \tag{75}$$

where G is the original graph, and G' is the sampled graph.

Degree structure. The second spectral moment is

$$\begin{aligned} m_2 &= \frac{1}{n}\operatorname{Tr}(P^2) \\ &= \frac{1}{n}\sum_i \sum_{j\in N(i)} \frac{1}{d_i d_j} \\ &= \frac{1}{n}\sum_i \mathbb{E}_{j\sim N(i)}\left[\frac{1}{d_j}\right]. \end{aligned} \tag{76}$$

Thus, m_2 measures the average inverse degree observed from the local neighborhood of each node.

More generally, let $X_0 = i$ be the starting node and X_r be the node reached after r random-walk steps. We define

$$L_i^{(r)} = \mathbb{E}\left[\frac{1}{d_{X_r}} \,\middle|\, X_0 = i\right] = \sum_j (P^r)_{ji} \frac{1}{d_j}. \tag{77}$$

The even-order spectral moment can be written as

$$\begin{aligned} m_{2r} &= \frac{1}{n} \operatorname{Tr}(P^{2r}) \\ &= \frac{1}{n} \sum_i \sum_j (P^r)_{ij} (P^r)_{ji} \\ &= \frac{1}{n} \sum_i d_i \sum_j \frac{(P^r)_{ji}^2}{d_j} \\ &= \frac{1}{n} \sum_i \widetilde{L}_i^{(r)}, \end{aligned} \tag{78}$$

where

$$\widetilde{L}_i^{(r)} = d_i \sum_j \frac{(P^r)_{ji}^2}{d_j}. \tag{79}$$

Compared with $L_i^{(r)}$, the quantity $\widetilde{L}_i^{(r)}$ reweights each endpoint contribution by the local accessibility factor $d_i (P^r)_{ji}$:

$$\widetilde{L}_i^{(r)} = \sum_j \left[d_i (P^r)_{ji}\right] \frac{(P^r)_{ji}}{d_j}. \tag{80}$$

For an r-step walk $\pi : i \overset{r}{\rightsquigarrow} j$, this factor admits the path expansion

$$d_i (P^r)_{ji} = \sum_{\pi : i \overset{r}{\rightsquigarrow} j} \prod_{v \in \mathrm{Int}(\pi)} \frac{1}{d_v}, \tag{81}$$

where $\mathrm{Int}(\pi)$ denotes the internal vertices of the walk. For example, when $r = 2$,

$$d_i (P^2)_{ji} = \sum_{k \in N(i) \cap N(j)} \frac{1}{d_k}. \tag{82}$$

Therefore, combining Eq. 81, Eq. 78 and Eq. 80, we can obtain Eq. 20:

$$m_{2r} = \frac{1}{n} \sum_i \mathbb{E}\left[\frac{a_i^{(r)}(X_r)}{d_{X_r}} \,\middle|\, X_0 = i\right], \quad a_i^{(r)}(j) = \sum_{\pi : i \overset{r}{\rightsquigarrow} j} \prod_{v \in \mathrm{Int}(\pi)} \frac{1}{d_v}. \tag{83}$$

C.2.2 Additional preservation results

Figure 6 provides additional results on ca-GrQc (Leskovec et al., 2007), Cora and Citeseer (Yang et al., 2016), and Wisconsin and Actor (Pei et al., 2020). Across these datasets, moment-preserving sampling consistently preserves the structural properties better than uniform random edge removal.

C.3 Moment-guided edge sampling for graph learning

This section provides additional experimental details for Section 6. We first examine how edges with different $\Delta\mathbf{m}(e)$ directions affect supervised node classification, and then evaluate moment-guided augmentation for unsupervised node classification.

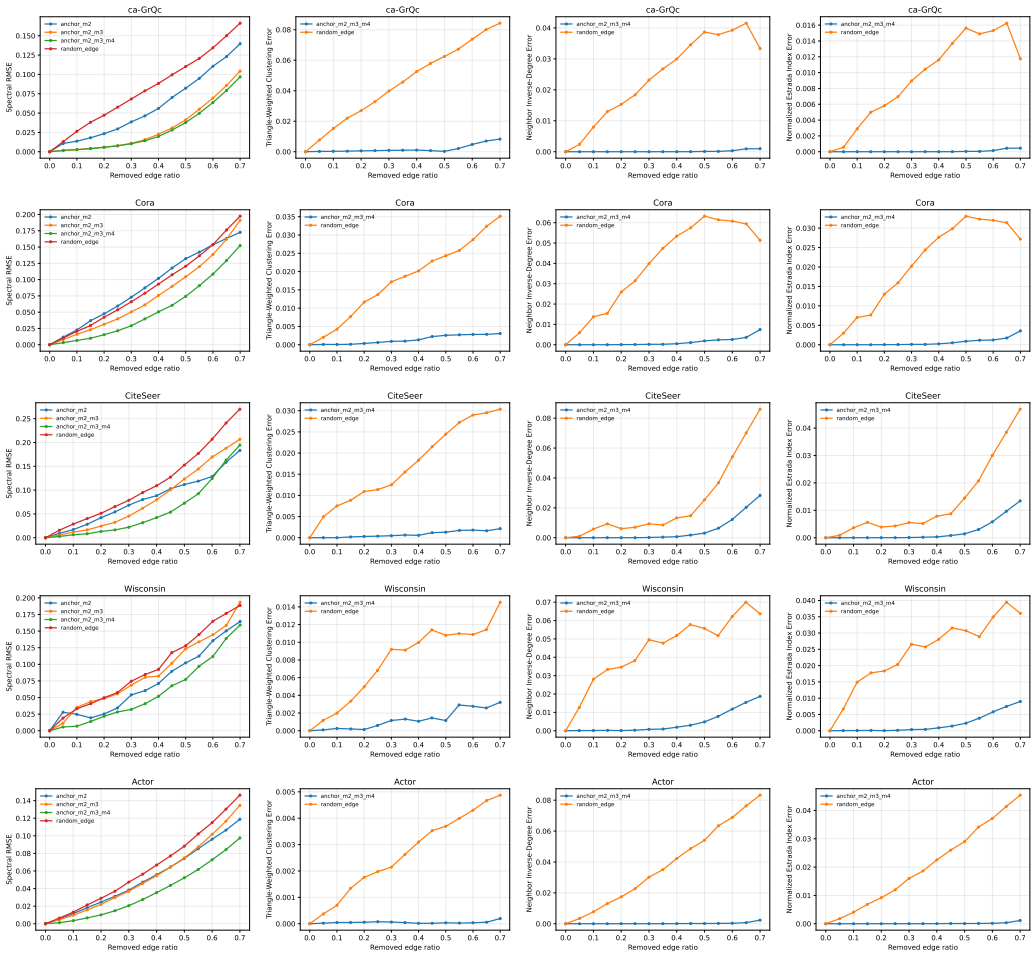


Figure 6: Moment-preserving sampling preserve four graph properties across datasets.

C.3.1 SUPERVISED NODE CLASSIFICATION

Experimental settings. We follow the "full" split (Chen et al., 2018), where all nodes outside the validation and test sets are used for training. During edge removal, we exclude edges whose endpoints are both training nodes with the same label. For each (direction, deletion ratio) pair, corresponding to one sector in Figure 4, we use five random seeds. For each seed, we sample five graphs using direction-specific edge removal and five graphs using uniform random removal under the same deletion budget. Each direction-specific sample is compared with its random counterpart to obtain five test-accuracy differences per seed. We report the mean over all $5 \times 5 = 25$ differences as the value of each sector.

C.3.2 GRAPH CONTRASTIVE LEARNING

Experimental settings. For the results in Table 1, we construct a moment-preserving augmented view using an edge-edit budget equal to 20% of the original number of edges, including both additions and deletions. Our implementation is built on the official GCL-SPAN code[1]. We replace its two augmented views with the original graph and our moment-preserving view, while keeping the remaining training pipeline unchanged. Following GCL-SPAN, we use a 10%/10%/80% train/validation/test split for the downstream classifier. The released implementation assigns the 80% split to validation, which we correct to match the stated protocol. Since dataset-specific hyper-

[1] https://github.com/Louise-LuLin/GCL-SPAN/tree/main

parameters are not fully provided, our reproduced GCL-SPAN results may differ from those reported in the original paper. All methods are evaluated over 10 runs using seeds 15–24.

We compare against three representative baselines:

- **GRACE** (Zhu et al., 2020) generates two views through random edge dropping. It treats the same node across views as a positive pair and all other nodes as negative pairs. We use the official implementation and hyperparameters[2].
- **MVGRL** (Hassani & Khasahmadi, 2020) contrasts representations obtained from the original graph and a diffusion-based view. We use the official implementation and reported hyperparameters[3]. Its PPR-based augmentation is also closely related to moments: Appendix A.2 shows that the trace of PPR can be written as a weighted sum of moments.
- **GCL-SPAN** (Lin et al., 2022): constructs augmented views by perturbing graph spectrum. Since spectral moments summarize the eigenvalue distribution, its augmentation is closely related to changing moments. We use the official implementation and tune the hyperparameters because dataset-specific settings are not provided. In contrast, targeting individual moments provides a more direct interpretation of which structural properties change, as discussed in Section 5 and Appendix C.2.

Searching the moment space for principled graph augmentation. Existing graph contrastive learning methods typically design augmentations through specific heuristics or implicit assumptions about which graph structures should be preserved or perturbed. For example, MVGRL relies on diffusion-based views, while GCL-SPAN explicitly perturbs graph spectrum. Such designs can be effective, but they do not directly reveal which structural changes are most beneficial for a given graph.

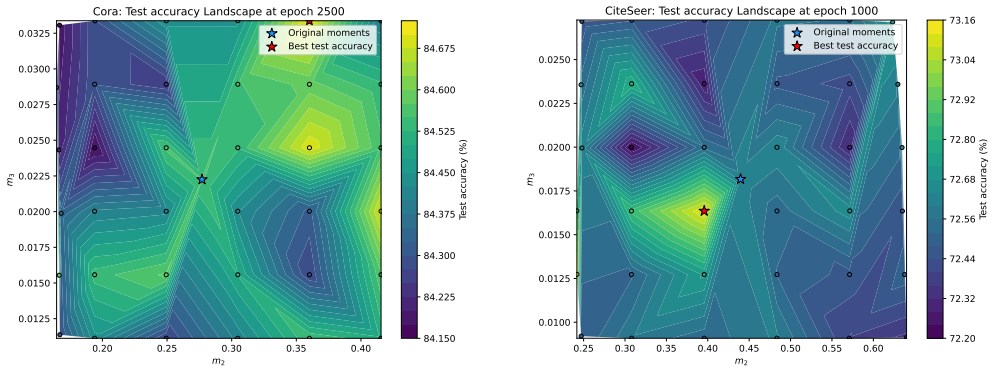
Spectral moments provide an explicit space for studying this question. Different target moments correspond to different normalized structural statistics, as discussed in Section 5. We therefore use the target moment profile as a controllable augmentation objective and systematically vary it to examine which structural changes lead to better representations.

Specifically, instead of always setting the target to the original moments, we sample 36 target profiles around the original (m_2, m_3) point. For each target, we edit 20% of the original number of edges and guide the graph toward that target using our moment-guided sampler (Section 3.3). Each target is evaluated over 10 random seeds, 15–24, and we report the mean test accuracy. This produces the accuracy landscapes shown in Figure 7.

The landscapes reveal *graph-specific augmentation preferences rather than a universal optimal rule*. For Cora, the best target lies around $(m_2, m_3) = (0.35, 0.033)$, where both m_2 and m_3 increase relative to the original graph. Since m_3 is closely related to triangle-weighted clustering, this direction tends to preserve or strengthen clustered local structure. In contrast, Citeseer performs best around $(m_2, m_3) = (0.40, 0.0163)$, where both moments decrease. The lower m_3 suggests that weakening clustered structure can benefit Citeseer, while the opposite trend is preferred by Cora. Thus, *different graphs favor different structural perturbations. The moment landscape makes these preferences explicit by mapping augmentation targets to interpretable structural changes, providing a principled way to search for graph-specific augmentation strategies*.

[2] https://github.com/CRIPAC-DIG/GRACE/tree/master

[3] https://github.com/kavehhassani/mvgrl



(a) Accuracy landscape on Cora.

(b) Accuracy landscape on Citeseer.

Figure 7: Accuracy landscapes over different target moment profiles. Each point corresponds to an augmented view generated toward a target (m_2, m_3) profile under the same edge-edit budget. The blue star marks the original moment profile, while the red star marks the target with the highest mean test accuracy.

Algorithm 1 Moment-Guided Edge Sampling

Require: Graph $G = (V, E)$, target moments $\mathbf{m}^*$, budget B, candidate size q, moment orders $\mathcal{K}$, edge-edit delta method DELTAMOMENT, which can be either COMBINATORIALDELTA or LOWRANKDELTA

Ensure: Sampled graph G_S

Compute current moments $\mathbf{m} = (m_k)_{k\in\mathcal{K}}$ ▷ Not necessarily continuous, such as $\mathcal{K} = 2, 5, 7$.
for each $k \in \mathcal{K}$ **do**
  $\sigma_k \leftarrow m_k$ ▷ Fixed to the original graph moment.
end for
for $t = 1, \ldots, B$ **do**
  Sample a candidate set $\mathcal{C}_t$ with $|\mathcal{C}_t| = q$, where each $\epsilon = (o, u, v) \in \mathcal{C}_t$ has $o \in \{\text{ADD}, \text{DELETE}\}$
  for each edge edit $\epsilon \in \mathcal{C}_t$ **do**
    $\Delta\mathbf{m}(\epsilon) \leftarrow \text{DELTAMOMENT}(G, \epsilon, \mathcal{K})$ ▷ Section 3.1 and 3.2
  end for
  for each edge edit $\epsilon \in \mathcal{C}_t$ **do**
    $\text{score}(\epsilon) \leftarrow \sum_{k\in\mathcal{K}} \left((m_k + \Delta m_k(\epsilon) - m_k^*)/\sigma_k\right)^2$ ▷ Section 3.3
  end for
  $\epsilon^\star \leftarrow \arg\min_{e\in\mathcal{C}_t} \text{score}(\epsilon)$
  APPLYOPERATION$(G, \epsilon^\star, \text{DELTAMOMENT})$
  $\mathbf{m} \leftarrow \mathbf{m} + \Delta\mathbf{m}(\epsilon^\star)$
end for
return $G_S \leftarrow G$
procedure APPLYOPERATION$(G, \epsilon, \text{DELTAMOMENT})$
  Write $\epsilon = (o, u, v)$
  if DELTAMOMENT $=$ COMBINATORIALDELTA **then** ▷ Update local statistics
    if $o =$ ADD **then**
      APPLYADD(u, v) ▷ Section B.2
    else
      APPLYDELETE(u, v) ▷ Section 3.1
    end if
  else ▷ Update graph for LOWRANKDELTA, Section 3.2
    Apply operation o on pair (u, v) to G
  end if
end procedure

Algorithm 2 COMBINATORIALDELTA: Exact Deltas for m_2 and m_3

Require: Graph G, candidate edge edit $\epsilon = (o, u, v)$, maintained statistics d_i, W_i, M_{ij}, H_i
Ensure: Exact moment changes $\Delta m_2(\epsilon), \Delta m_3(\epsilon)$

if $o =$ ADD **then**
  $\Delta S_2 \leftarrow \dfrac{1}{(d_u+1)(d_v+1)} - \dfrac{W_u}{d_u(d_u+1)} - \dfrac{W_v}{d_v(d_v+1)}$ ▷ Eq. 26
  $\Delta S_3 \leftarrow \dfrac{M_{uv}}{(d_u+1)(d_v+1)} - \dfrac{H_u}{d_u(d_u+1)} - \dfrac{H_v}{d_v(d_v+1)}$ ▷ Eq. 27
else if $o =$ DELETE **then**
  $\Delta S_2 \leftarrow \dfrac{W_u - d_v^{-1}}{d_u(d_u-1)} + \dfrac{W_v - d_u^{-1}}{d_v(d_v-1)} - \dfrac{1}{d_u d_v}$ ▷ Eq. 4
  $\Delta S_3 \leftarrow \dfrac{H_u - M_{uv}/d_v}{d_u(d_u-1)} + \dfrac{H_v - M_{uv}/d_u}{d_v(d_v-1)} - \dfrac{M_{uv}}{d_u d_v}$ ▷ Eq. 5
end if
return $\left(\dfrac{2}{n}\Delta S_2, \dfrac{6}{n}\Delta S_3\right)$

Algorithm 3 APPLYADD: Update Local Statistics for COMBINATORIALDELTA

procedure APPLYADD(u, v)
  $N_u \leftarrow \mathcal{N}(u), N_v \leftarrow \mathcal{N}(v)$
  Use old degrees d_u, d_v ▷ Update degrees after finish updating all local statistics
  $\Delta H \leftarrow$ DELTAHADD(u, v)
  for all $x \in N_u$ **do** ▷ Update the local statistics between u's neighbors and v
    $M_{vx}, M_{xv} \leftarrow M_{vx} + (d_u+1)^{-1}, M_{xv} + (d_u+1)^{-1}$ ▷ Edge addition introduce a new common neighbor u between v and u's neighbors
    $W_x \leftarrow W_x - \big(d_u(d_u+1)\big)^{-1}$ ▷ The increase in u's degree dilutes the weights assigned to its neighbors'
  end for
  for all $x \in N_v$ **do** ▷ Update the local statistics between v's neighbors and u
    $M_{ux}, M_{xu} \leftarrow M_{ux} + (d_v+1)^{-1}, M_{xu} + (d_v+1)^{-1}$ ▷ Edge addition introduce a new common neighbor v between u and v's neighbors
    $W_x \leftarrow W_x - \big(d_v(d_v+1)\big)^{-1}$ ▷ The increase in v's degree dilutes the weights assigned to its neighbors'
  end for
  for all $x, y \in N_u$ **do** ▷ Update the local statistics within u's neighbors
    $M_{xy} \leftarrow M_{xy} - \big(d_u(d_u+1)\big)^{-1}$ ▷ The increase in u's degree dilutes the weights assigned to its neighbors'
  end for
  for all $x, y \in N_v$ **do** ▷ Update the local statistics within v's neighbors
    $M_{xy} \leftarrow M_{xy} - \big(d_v(d_v+1)\big)^{-1}$ ▷ The increase in v's degree dilutes the weights assigned to its neighbors'
  end for
  Add $\{u, v\}$ to E
  $\mathcal{N}(u) \leftarrow N_u \cup \{v\}, \mathcal{N}(v) \leftarrow N_v \cup \{u\}$
  $d_u \leftarrow d_u + 1, d_v \leftarrow d_v + 1$
  $W_u \leftarrow W_u + d_v^{-1}, W_v \leftarrow W_v + d_u^{-1}$ ▷ Edge addition introduce a new term for u and v, after updating degrees
  for all i with $\Delta H_i \neq 0$ **do**
    $H_i \leftarrow H_i + \Delta H_i$
  end for
end procedure

Algorithm 4 APPLYDELETE: Update Local Statistics for COMBINATORIALDELTA

procedure APPLYDELETE(u, v)
  $N_u \leftarrow \mathcal{N}(u)$, $N_v \leftarrow \mathcal{N}(v)$
  Use old degrees d_u, d_v
  $N_u^- \leftarrow N_u \setminus \{v\}$, $N_v^- \leftarrow N_v \setminus \{u\}$
  $\Delta H \leftarrow$ DELTAHDELETE(u, v)
  for all $x \in N_u^-$ **do** ▷ Update the local statistics between u's neighbors and v
    $M_{vx}, M_{xv} \leftarrow M_{vx} - d_u^{-1}, M_{xv} - d_u^{-1}$ ▷ Edge deletion remove the common neighbor u between v and u's neighbors
  end for
  for all $x \in N_v^-$ **do** ▷ Update the local statistics between v's neighbors and u
    $M_{ux}, M_{xu} \leftarrow M_{ux} - d_v^{-1}, M_{xu} - d_v^{-1}$ ▷ Edge deletion remove the common neighbor v between u and v's neighbors
  end for
  for all $x, y \in N_u^-$ **do** ▷ Update the local statistics within u's neighbors
    $M_{xy} \leftarrow M_{xy} + \big(d_u(d_u - 1)\big)^{-1}$ ▷ The decrease in u's degree concentrate the weights assigned to its neighbors'
  end for
  for all $x, y \in N_v^-$ **do** ▷ Update the local statistics within u's neighbors
    $M_{xy} \leftarrow M_{xy} + \big(d_v(d_v - 1)\big)^{-1}$ ▷ The decrease in v's degree concentrate the weights assigned to its neighbors'
  end for
  $W_u \leftarrow W_u - d_v^{-1}, W_v \leftarrow W_v - d_u^{-1}$ ▷ Edge deletion remove one term for u and v, before updating degrees
  Delete $\{u, v\}$ from E
  $\mathcal{N}(u) \leftarrow N_u^-, \mathcal{N}(v) \leftarrow N_v^-$
  $d_u \leftarrow d_u - 1, d_v \leftarrow d_v - 1$
  for all $x \in N_u^-$ **do** ▷ Edge deletion concentrate weights assigned to u's neighbors
    $W_x \leftarrow W_x + \big((d_u + 1)d_u\big)^{-1}$
  end for
  for all $x \in N_v^-$ **do** ▷ Edge deletion concentrate weights assigned to v's neighbors
    $W_x \leftarrow W_x + \big((d_v + 1)d_v\big)^{-1}$
  end for
  for all i with $\Delta H_i \neq 0$ **do**
    $H_i \leftarrow H_i + \Delta H_i$
  end for
end procedure

Algorithm 5 Exact Local Update of H

function WCOMMON($a, b, \mathcal{R}$) ▷ Weighted common-neighbor mass, excluding nodes in $\mathcal{R}$
  $s \leftarrow 0$
  for all $x \in \mathcal{N}(a) \cap \mathcal{N}(b)$ **do**
    if $x \notin \mathcal{R}$ **then**
      $s \leftarrow s + d_x^{-1}$
    end if
  end for
  return s
end function
function DELTAHADD(u, v)
  Initialize $\Delta H_i \leftarrow 0$ for $i \in \{u, v\} \cup \mathcal{N}(u) \cup \mathcal{N}(v)$
  $C_{uv} \leftarrow$ WCOMMON$(u, v, \emptyset)$ ▷ New triangles centered at u and v
  $\Delta H_u \leftarrow \Delta H_u + C_{uv}/(d_v + 1)$
  $\Delta H_v \leftarrow \Delta H_v + C_{uv}/(d_u + 1)$
  for all $i \in \mathcal{N}(u)$ **do** ▷ Old triangles containing u diluted because d_u increases
    $C_{iu} \leftarrow$ WCOMMON$(i, u, \emptyset)$
    $\Delta H_i \leftarrow \Delta H_i - C_{iu}/\big(d_u(d_u + 1)\big)$
  end for
  for all $i \in \mathcal{N}(v)$ **do** ▷ Old triangles containing v diluted because d_v increases
    $C_{iv} \leftarrow$ WCOMMON$(i, v, \emptyset)$
    $\Delta H_i \leftarrow \Delta H_i - C_{iv}/\big(d_v(d_v + 1)\big)$
  end for
  for all $i \in \mathcal{N}(u) \cap \mathcal{N}(v)$ **do** ▷ New triangle (i, u, v) for each common neighbor i
    $\Delta H_i \leftarrow \Delta H_i + \big((d_u + 1)(d_v + 1)\big)^{-1}$
  end for
  return ΔH
end function
function DELTAHDELETE(u, v)
  Initialize $\Delta H_i \leftarrow 0$ for $i \in \{u, v\} \cup \mathcal{N}(u) \cup \mathcal{N}(v)$
  $C_{uv} \leftarrow$ WCOMMON$(u, v, \emptyset)$ ▷ Triangles centered at u and v using edge (u, v) disappear
  $\Delta H_u \leftarrow \Delta H_u - C_{uv}/d_v$
  $\Delta H_v \leftarrow \Delta H_v - C_{uv}/d_u$
  for all $i \in \mathcal{N}(u) \setminus \{v\}$ **do** ▷ Remaining old triangles containing u grow because d_u decreases
    $C_{iu}^{-v} \leftarrow$ WCOMMON$(i, u, \{v\})$
    $\Delta H_i \leftarrow \Delta H_i + C_{iu}^{-v}/\big(d_u(d_u - 1)\big)$
  end for
  for all $i \in \mathcal{N}(v) \setminus \{u\}$ **do** ▷ Remaining old triangles containing v grow because d_v decreases
    $C_{iv}^{-u} \leftarrow$ WCOMMON$(i, v, \{u\})$
    $\Delta H_i \leftarrow \Delta H_i + C_{iv}^{-u}/\big(d_v(d_v - 1)\big)$
  end for
  for all $i \in \mathcal{N}(u) \cap \mathcal{N}(v)$ **do** ▷ Triangle (i, u, v) disappears for each common neighbor i
    $\Delta H_i \leftarrow \Delta H_i - (d_u d_v)^{-1}$
  end for
  return ΔH
end function

Algorithm 6 LOWRANKDELTA: Exact Moment Delta for Batched Edge Edits, see Section 3.2

Require: Graph $G = (V, E)$, transition matrix $P = AD^{-1}$, edit batch $\mathcal{B}$, moment orders $\mathcal{K}$
Ensure: Exact changes $\{\Delta m_k\}_{k\in\mathcal{K}}$

$T \leftarrow \{i \in V : \exists j \in V \text{ such that } (i, j) \in \mathcal{B}\}$
Write $T = \{i_1, \ldots, i_r\}$
Compute updated touched columns $P'_{:i_a}$ for all $i_a \in T$
Initialize $C \in \mathbb{R}^{n\times r}$ and $R \in \mathbb{R}^{n\times r}$
for $a = 1, \ldots, r$ **do**
    $C_{:a} \leftarrow P'_{:i_a} - P_{:i_a}$
    $R_{:a} \leftarrow e_{i_a}$
end for
$K_{\max} \leftarrow \max \mathcal{K}$
$Y \leftarrow C$
for $t = 0, \ldots, K_{\max} - 1$ **do**
    $H_t \leftarrow R^\top Y$
    if $t < K_{\max} - 1$ **then**
        $Y \leftarrow PY$
    end if
end for
for each $k \in \mathcal{K}$ **do**
    $\Delta T_k \leftarrow$ TRACEDELTA$(H_0, \ldots, H_{k-1}, k)$
    $\Delta m_k \leftarrow \Delta T_k / n$
end for
return $\{\Delta m_k\}_{k\in\mathcal{K}}$

Algorithm 7 TRACEDELTA: Trace-Power Change from Small Matrices, see Section 3.2

Require: Small matrices $H_0, \ldots, H_{k-1}$, moment order k
Ensure: Exact trace-power change ΔT_k

$\Delta T_k \leftarrow 0$
for $\ell = 1, 2, \ldots, k$ **do**
    for each $(a_1, \ldots, a_\ell) \in \mathcal{A}_{k-\ell,\ell}$ **do**
        $M \leftarrow I$
        for $q = 1, \ldots, \ell$ **do**
            $M \leftarrow MH_{a_q}$
        end for
        $\Delta T_k \leftarrow \Delta T_k + \frac{k}{\ell} \operatorname{Tr}(M)$
    end for
end for
return ΔT_k