Merge-Split: Directed Graph Perturbations that Preserve Random Walk Structure (Flavor: Systems)

Sachin Konan
Two Sigma Investments, LP
New York, New York
sachin@twosigma.com

Larry Rudolph
Two Sigma Investments, LP
New York, New York
larry.rudolph@twosigma.com

ABSTRACT

Directed graphs contain valuable information about the relationships between entities, and whose structure can reveal a multitude of dependencies. This makes directed graphs ideal datasets for analysis, but their exact structural topology might be sensitive and in need of protection. Previous literature focuses on protecting the privacy of undirected, power-law social network graphs, but given the proliferation of data pipelines and Directed Acyclic Graphs, we focus on *directed, lineage* graphs. We propose Merge-Split, a graph perturbation algorithm that increasingly obscures subgraph structural identity while minimizing overall general changes in structure. Merge-Split perturbs the graph with a sequence of vertex merges and splits, whose order is determined by a potential function that minimizes the change in a graph's spectral decomposition. Restricted Spectral Similarity (RSS) has been used to compute the spectral change induced from coarsening undirected graphs, and we extend this by providing *Directed*-RSS, which computes the spectral distance between a directed graph and its Merge-Split-perturbed counterpart. In addition, the disruption and preservation of local and global random walks, respectively, are measured by introducing a *Random Walk* KL Divergence metric. Results indicate that our potential function optimizes both across 4 test graphs.

PVLDB Reference Format:
Sachin Konan and Larry Rudolph. Merge-Split: Directed Graph Perturbations that Preserve Random Walk Structure (Flavor: Systems). PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

1 INTRODUCTION

The realization that lineage graphs, or directed acyclic graphs (DAGs) in general, contain valuable information implies that the valuable information needs some type of protection, especially when they are shared or published. There are many uses of lineage, including governance, dependency analysis, data discovery, observability and orchestration. Different attributes of the graph are relevant to each of these use cases. We propose Merge-Split as a way to perturb a lineage graph whose overall structure and lineage are maintained while its local structure is sufficiently obscured to protect their value. Differential privacy can be used to obscure numeric attributes of the nodes and edges, however, deleting or adding nodes and edges can destroy the lineage information as well as global structure. Rather than adding noise to the graph, Merge-Split perturbs the graph in a way that minimizes the change in its spectrum.

The consideration of privacy in the publication of graphs has largely been driven by efforts to deanonymize them. Backstrom et. al [2] describe attacks that enable the identification of $O(N^2)$ nodes from $O(logN)$ "Sybil node" insertions. Narayan et. al [32] extended this work by introducing the clique-isomorphism-search attack used to deanonymize users in social networks. Differential Privacy (DP) [8] partially addressed these concerns by protecting individual node/edge privacy during aggregative undirected graph queries such as [18], subgraph count (such as the number of k-cliques) [14, 16] and degree distribution [7, 14, 17, 18]. However, to release the graphs themselves, researchers have worked on synthetic graph generation [40, 47] and perturbation[12, 22].

These works focus on *undirected*, *power-law* graphs, and although many studies support their ubiquity [10, 29, 36], others challenge them [11, 21, 39]. Broido et. al [3] found that amongst a large diversity of network datasets, strongly power-law graphs are empirically rare, while for most networks, log-normal distributions better fit the data.

Directed, non-power law graphs are characteristic of an increasingly common set of graphs in today's data ecosystem: DAGs and Data Pipelines. DAGs capture acyclic, causal dependencies, and tend to follow non-power-law distributions as there is rarely one node that is a causal dependency for every other node in the graph [38]. DAGs are universally used in data center scheduling algorithms [9, 24], epidemiology factor analysis [38], hierarchical knowledge graphs [5, 26], or in the computation execution of neural networks [37, 45]. The machine learning community can benefit from the publication of productive DAGs but without revealing their "secret sauce." In general directed, lineage-like graphs can be used to create better schedulers [1, 19] or to model complex causal relationships [6, 34]. Additionally, almost all modern-day companies rely upon data pipelines to ingest and process data at scale [27, 46], and internally, companies can benefit from the dispersion of these pipelines to allow engineers to diagnose failure modes or inefficiencies better [24]. Since the key aspect of these graphs is their structural organization, or the ancestral lineage of each node, we refer to these graphs as *lineage* graphs. There is tremendous utility in releasing *lineage* graphs to the Machine Learning community and within corporations, although privacy leakage remains a central concern.

We propose Merge-Split, a DAG perturbation algorithm that generates publishable lineage graphs by selectively merging and splitting nodes (and their corresponding edge sets). We show that

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit https://creativecommons.org/licenses/by-nc-nd/4.0/ to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.
Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

iterative applications of Merge-Split increasingly disturb the type-/frequency of local subgraphs, making it difficult to identify a graph's real/obfuscated components. Besides directedness, Merge-Split differs from previous works because it maintains overall structure in the context of directed random walks by minimizing the change in the graph's spectrum. Our contributions are:

(1) Merge-Split: a directed graph perturbation algorithm, that has the ability to obfuscate nodes/edges of the graph while maintaining overall structural properties in the context of lineage (i.e. directed random walks).
(2) Efficient and exact calculation of Merge-Split's induce spectral change, which we term: *Directed Restricted Spectral Similarity*. We show that this property upper-bounds the change in a graph's steady state distribution.
(3) *Random-Walk KL Divergence* is used to measure local/global structural differences. This metric could be a direct optimization objective of learning-based perturbation methods.
(4) A potential function that samples operations that empirically minimize the Aggregate Directed Restricted Spectral Similarity on four directed graphs while perturbing local substructures as captured by Random-Walk KL Divergence.
(5) Empirical results that show that iterative applications of Merge-Split increasingly disturb the frequency and type of subgraphs present between the original and perturbed graphs which justify our conjecture that Merge-Split increases the difficulty of locating subgraphs.

The rest of the paper is organized as follows. Section 2 provides the foundations and basic definitions, including what is meant by lineage preservation and the basic merge and split operations. Section 3 introduces our spectral similarity metric for directed graphs that is used to control which operations are performed with which nodes. Empirical verification of this metric is presented in Section 4. The heart of the algorithm is in Section 5, while Sections 6 and 7 show that the properties are preserved. Finally, we show the similarities and differences between our work and published results and conclude with a visual representation of the effects of various perturbation choices.

2 FOUNDATIONS

We seek to publish lineage graphs (i.e. directed, non-power law) with the lineage information and global structure preserved while local structure is obscured. Lineage graphs generally capture complex dependencies (e.g. edges) on raw input objects (e.g. source nodes), a series of transformation hierarchy (e.g. internal nodes) and a set of output (e.g. sink nodes).

Local structure is obscured by *perturbing* the original graph. Perturbation is the process by which the nodes/edges of a graph are "perturbed" with randomized creation/deletion, allowing unaffected nodes/edges to retain their identity. It should come as no surprise that we do this by a sequence of merge and splits.

2.1 Lineage Preservation

Definition 1. **Lineage Preservation**: *Let $\mathbb{D}$ be the set of vertices that are part of the process of* perturbing *graph G into graph G' and for all vertices u not in $\mathbb{D}$, there is a 1-1 correspondence with nodes in G'.Lineage is preserved when if there is a path in G from u to v, then there is a path in G' from u' to v', where u', v' correspond to u, v.*

$$\forall_{(u,v)\notin\mathbb{D}} \quad u \to \ldots \to v \implies u' \to \ldots \to v' \tag{1}$$

2.2 Merge and Splits on Graphs

The goal of Merge-Split is to perturb a directed graph $G = (\mathbb{V}, \mathbb{E})$, with vertex set $\mathbb{V}$ and edge set $\mathbb{E}$ where $e(u, v) \in \mathbb{E}$ and u is source and v is destination. G can be represented by adjacency matrix (A),with each edge's initial value is 1. The value at any index (u, v) is the edge's weight $w_{u,v}$. Let v's predecessor set (e_v^{in}), successor (e_v^{out}) set, and degree of a vertex d_v be defined as:

$$e_v^{in} = \{(u, v) \in \mathbb{E}\} \qquad e_v^{out} = \{(v, u) \in \mathbb{E}\} \qquad d_v = \sum_{(v,u)\in e_v^{out}} w_{v,u}$$

Our operations for perturbation: *Merge* and *Split* are as follows:

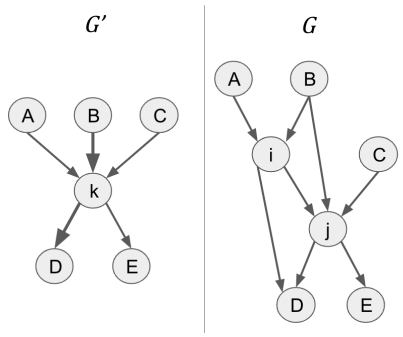

Figure 1: Merge Operation: $\mathbb{M}_{i,j\to k}(G) = G'$. Darkened arrows indicate the combination of unit edge weights.

Merge Operation: A merge, $\mathbb{M}_{i,j\to k}(G) = G'$, is an atomic operation that combines any two directly connected nodes i, j into a composite node k, where e_k^{in} is the union of the predecessors of i, j connected to k and e_k^{out} is the union of the successors of i, j originating from k (see Figure 1). The weights of the original edges are copied (or summed for duplicate predecessors or successors of i and j) to retain the degree of the nodes in the graph, so the total change in weight after a merge is the loss of one edge, $-w_{i,j}$. This operation creates a new graph $G' = (\mathbb{V}', \mathbb{E}')$:

$$\mathbb{V}' = \mathbb{V} - \{i, j\} + \{k\} \quad \mathbb{E}' = \mathbb{E} - \{\forall_{v\in\{i,j\}} e_v^{in} \cup e_v^{out}\} + \{e_k^{in} \cup e_k^{out}\} \tag{2}$$

Formally, a merge is a surjective mapping between $\mathbb{V} \to \mathbb{V}'$, and if $|\mathbb{V}| = N$, then $|\mathbb{V}|' = N - 1$. A dimension-reducing matrix $C_{\mathbb{M}} \in R^{N-1\times N}$ describes how $v \in \mathbb{V}$ is mapped onto the $\mathbb{V}'$. Thus if there exists a vector $x \in R^N$, its merged equivalent is $C_{\mathbb{M}}x$, whose matrix is defined as:

$$C_{\mathbb{M}}(x, y) = \begin{cases} 1/\sqrt{2} & \text{if } x = k \ \ \& \ \ (y = i \vee y = j) \\ 1 & \text{if } x = y \ \ \& \ \ (x \neq i \ \ \& \ \ y \neq j) \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

It's important to note that $C_{\mathbb{M}}$ is an orthonormal matrix, and thus $C_{\mathbb{M}}C_{\mathbb{M}}^T = I$, which is a useful property in the definition of spectral similarity. Additionally, *merges are always lineage-preserving* because, in G', there are always paths from the predecessors of i, j to their successors.

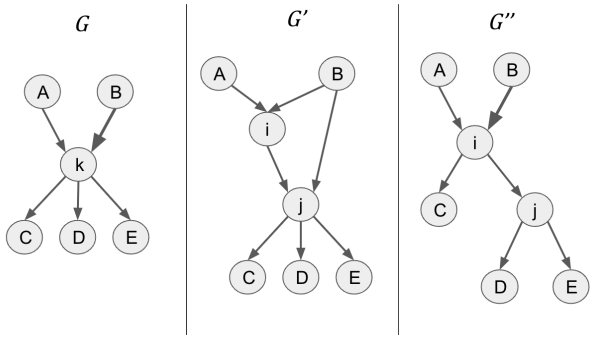

Figure 2: Split Operation: $\mathbb{S}_{k\to i,j}(G) = G'$ **or** $split funcG = G''$**. Lightened arrows indicate the splitting of unit edge weights.**

Split Operation: A split, $\mathbb{S}_{k\to i,j}(G)$, is an atomic operation that splits a node k into two nodes i, j that are conjoined by an edge (i, j), with $w_{i,j} = 1$, and allocates the predecessor/successor sets of k across i, j. This allocation is key for preserving lineage. Therefore, we utilize two non-negative functions $\psi^{in}(), \psi^{out}() \geq 0$, where $\psi_v^{in}(w_{u,k}) = w_{u,v}$ allocates some amount of $w_{u,k}$ to the newly created edge (u, v) and $\psi_v^{out}(w_{k,u}) = w_{v,u}$ allocates some amount of $w_{k,u}$ to the newly created edges (v, u). For the vertices, i and j, the edge partition functions are constrained such that:

$$\forall_{(u,k)\in e_k^{in}} \ (\psi_i^{in}(w_{u,k}) + \psi_j^{in}(w_{u,k}) = w_{u,k})$$
$$\forall_{(k,u)\in e_k^{out}} \ (\psi_i^{out}(w_{k,u}) + \psi_j^{out}(w_{k,u}) = w_{k,u})$$

Under this condition, the weight of the graph increases by $w_{i,j} = 1$. The edges in the input/output edge sets of i, j are defined whenever $\psi()$ returns a positive weight, so the vertex and edge sets of G':

$$\mathbb{V}' = \mathbb{V}+\{i, j\}-\{k\} \quad \mathbb{E}' = \mathbb{E}+\{\forall_{v\in\{i,j\}} e_v^{in}\cup e_v^{out}\}-\{e_k^{in}\cup e_k^{out}\} \quad (4)$$

A split can be thought of as an injective mapping between $\mathbb{V} \to \mathbb{V}'$ where $|\mathbb{V}|' = N + 1$. We define a dimension-increasing split matrix $C_{\mathbb{S}} \in R^{N+1\times N}$ that describes how each $v \in \mathbb{V}$ is mapped onto $\mathbb{V}'$. This matrix is exactly the transpose of $C_{\mathbb{M}}$, and thus it follows its orthonormal properties.

For lineage, it is necessary that predecessors of k have a path to successors of k. There is a large domain of $\psi^{in}(), \psi^{out}()$ that fits this criterion, so to reduce a split's complexity, we define two specific instances of lineage-maintaining splits: $\mathbb{S}_{k\to i,j}^{in}(k), \mathbb{S}_{k\to i,j}^{out}(k)$. $\mathbb{S}_{k\to i,j}^{in}(k)$ creates G' as seen in Fig. 2, where $\psi_i^{in}(), \psi_j^{in}()$ randomly allocate k's predecessor weights amongst i, j, while $\psi_j^{out}()$ allocates all of k's successor weights to j. $\mathbb{S}_{k\to i,j}^{out}(k)$ creates G'' as seen in Fig. 2, where $\psi_i^{out}(), \psi_j^{out}()$ randomly allocate the successor weights of k amongst i, j such that $d_i = d_j = d_k/2$, while $\psi_i^{in}()$ allocates all predecessors of k to i. Therefore, $\mathbb{S}_{k\to i,j}^{in}(k), \mathbb{S}_{k\to i,j}^{out}(k)$ *are always lineage-preserving.*

3 SPECTRAL SIMILARITY FOR DIRECTED GRAPHS

Merge-Split obfuscates G's structure, but by how much? Loukas et. al [23] (extending Spielman et. al [43, 44]) define Restricted Spectral Similarity (RSS) which is the change in the quadratic form of a graph's Laplacian (L) with respect to its coarsened-counterpart, over the eigenspace of L's first K eigenvectors. They show that minimizing RSS leads to the maintenance of the graph's clustering properties and spectrum, which previous literature connected to holistic measures of graph structure, e.g. diameter or conductance [28]. Since minimizing RSS is tied to the preservation of overall structural properties, Merge-Split minimizes RSS to minimize the structural difference between G and G'. For *directed* graphs:

Definition 2. ***Random Walk Normalized Laplacian****: $L_{RW} = I - D^{-1}A$, where D is the diagonal matrix, such that $D[i, i] = d_i$ and A is the adjacency matrix (defined in Sec.2.2). The eigenvalue/eigenvector pairs of L_{RW} are represented as (λ_i, x_i), where x_i is the eigenvector corresponding to the i-th eigenvalue; however, unlike the undirected Laplacian, these pairs may be complex.*

L_{RW} cannot be substituted into the definition of RSS, because Loukas et. al[23] not only assumes that the graphs are undirected, but also that there exists a linear transformation that can transform L into its coarsened counterpart. In our case, there doesn't exist a linear transformation that can convert L_{RW} into its Merge-Split-perturbed counterpart, because $\mathbb{S}_{k\to i,j}$ is non-deterministic. Therefore, we define a notion of RSS for Directed Graphs that are perturbed with Merge-Split in Eq. 5.

Definition 3. ***Directed-RSS****: $\vec{\delta} \in \mathbb{C}^K$ is a measure of how similar two graphs are with respect to random walks. After performing a merge or split, $\vec{\delta}$ captures the change in the quadratic form of G's Random-Walk Laplacian (L_{RW}) to that of G' (L'_{RW}). Let $C \in \{C_{\mathbb{M}}, C_{\mathbb{S}}\}$ be the dimension-shifting matrix to adjust the dimension of the n-th eigenvector ($\vec{x_n}$) of L_{RW} to the dimension of L'_{RW}. Let Directed-RSS be defined as follows, where we use bracket notation to refer to a particular index of a vector and $x'_n = Cx_n$:*

$$\forall_{1\leq n\leq K} \quad \vec{\delta}[n] = (Cx_n)^T L'_{RW}(Cx_n) - x_n^T L_{RW} x_n \quad (5)$$
$$= \sum_{(u,v)\in\mathbb{E}'} (1/d_u) w_{u,v} x'_n[u](x'_n[u] - x'_n[v])$$
$$- \sum_{(u,v)\in\mathbb{E}} (1/d_u) w_{u,v} x_n[u](x_n[u] - x_n[v]) \quad (6)$$

$\vec{\delta}_{\mathbb{O}}$ is the similarity measure between the graphs where one general operation $\mathbb{O}$ is performed, where $\mathbb{O} \in \{\mathbb{M}_{i,j\to k}, \mathbb{S}_{k\to i,j}\}$.

As seen in Eq. 6, each calculation of $\vec{\delta}$ requires $O(|\mathbb{E}|)$, which is expensive for the entire domain of merges and splits. However, since Merge-Split only affects a local neighborhood of vertices per iteration, the Directed-RSS is efficiently computed from just those vertices incident to operation. For the Directed-RSS Eq. 7 is the simplified version for merge and Eq. 8 is the *expected* value for a split (based on a random choice of $\mathbb{S}_{k\to i,j}^{in}(k), \mathbb{S}_{k\to i,j}^{out}(k)$). For brevity, we use $\mathbb{M}_{i,j}, \mathbb{S}_k$ for $\mathbb{M}_{i,j\to k}, \mathbb{S}_{k\to i,j}$. It takes $O(d_{\max})$ to compute $\vec{\delta}_{\mathbb{M}_{i,j}}$ and $\vec{\delta}_{\mathbb{S}_k}$. Therefore, the calculation across all merge and splits, and the first K eigenvectors, is $O(K(|\mathbb{V}| + |\mathbb{E}|)d_{\max})$, which need only

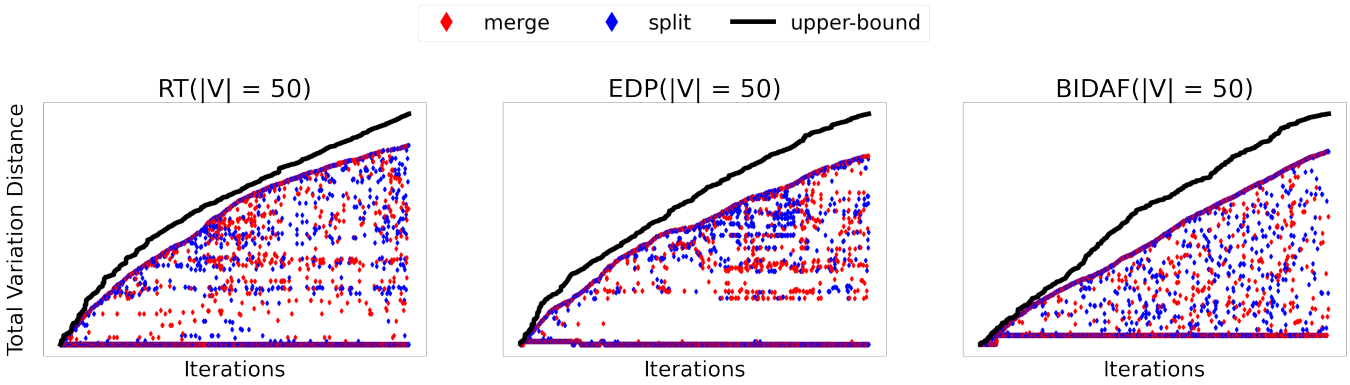

Figure 3: Instantaneous Total Variation Distance (Δy_{max}) of x_s on RT, EDP, and BIDAF.

be calculated once (Sec. 5). Here $d_{\max}$ = maximum degree.

$$\vec{\delta}_{\mathbb{M}_{i,j}}[n] = \frac{3w_{i,j}x_n[i](x_n[j]-x_n[i])}{d_i} + \sum_{t\in\{i,j\}}\left[\sum_{(u,t)\in e_t^{in}} \frac{w_{u,t}x_n[u]}{d_u}(x_n[t]-x'_n[k]) + \sum_{(t,u)\in e_t^{out}} w_{t,u}\left(\frac{x'_n[k](x'_n[k]-x_n[u])}{d_k} - \frac{x_n[t](x_n[t]-x_n[u])}{d_t}\right)\right] \tag{7}$$

$$\mathbb{E}[\vec{\delta}_{\mathbb{S}_k}[n]] = \sum_{(u,k)\in e_k^{in}} \frac{w_{u,k}x_n[u]x_n[k]}{d_u}\left(1-\frac{1}{\sqrt{2}}\right) + \sum_{(k,u)\in e_k^{out}} \frac{w_{u,k}x_n[k]}{d_u}\left(\frac{2\sqrt{2}-3}{2\sqrt{2}}x_n[u] - \frac{x_n[k]}{4}\right) \tag{8}$$

DRSS and Steady State: L_{RW} is related to the transition probability matrix (P) of G, since $L_{RW} = I - P$. It shares the eigenspace of P, (i.e. if λ_i is an eigenvalue of P, then $1-\lambda_i$ is an eigenvalue of L_{RW}). For graphs with absorbing states, (i.e. DAGs, pipelines, and trees), there exists at least one stationary distribution, $x_s \in R^{|\mathbb{V}|}$, such that $x_s^T = x_s^T P = x_s^T(I - L_{RW})$. If there exists an x_s, then $\lambda_s = 1$ is an eigenvalue of P and $\lambda_s = 0$ must be an eigenvalue of L_{RW} with eigenvector x_s. A graph's stationary distribution is the distribution across vertices for an infinite-length random walk, and the differences in a graph's structure can be conveyed through changes in its stationary distribution [30]. Thus, the relationship between Directed-RSS and x_s is used to establish a connection between Directed-RSS's ability to measure the structural change in the context of random walks.

Consider a merge or split operation applied on G. To compute the one-step random walk, on L'_{RW} using x_s, x_s must be transformed to the same dimension of L'_{RW} which can be done using $C \in \{C_{\mathbb{M}}, C_{\mathbb{S}}\}$. However, C is an orthonormal matrix that doesn't preserve stochasticity, so it is necessary to introduce new variables $\xi_1, \xi_2 \in dim(Cx_s)$, such that $Cx_s + \xi_2$ is stochastic (Details of which can be found in the Appendix). We compare the one-step random walk of the initial distribution x_s on L'_{RW}, call this $y_2^T = (Cx_s + \xi_2)^T(I - L')$, to the transformed, stochastic version of the steady-state $y_1^T = (Cx_s + \xi_1)^T$, with $\Delta^T y = y_2^T - y_1^T$. Δy_{max} measures the total variation distance between y_2^T, y_1^T, which we bound for merges and splits in Eq. 9 and Eq. 10, respectively.

$$\mathbb{M}_{i,j} : \Delta y_{max} \leq \frac{\vec{\delta}_{\mathbb{M}_{i,j}}[s] - \xi_x^T C_{\mathbb{M}} x_s}{1 - \frac{(\sqrt{2}-1)(x_s[i]+x_s[j])}{\sqrt{2}}} \tag{9}$$

$$\mathbb{S}_k : \Delta y_{max} \leq \frac{\vec{\delta}_{\mathbb{S}_k}[s] - \xi_s^T C_{\mathbb{S}} x_s}{1 - \frac{\sqrt{2}-2}{\sqrt{2}}x[k]} \tag{10}$$

These bounds imply that minimizing $\vec{\delta}_{\mathbb{M}_{i,j}}[s]$ and $\vec{\delta}_{\mathbb{S}_k}[s]$ (the Directed-RSS on x_s) also minimize Δy_{max}.

4 EMPIRICAL VERIFICATION

Throughout this paper, we present empirical results to both verify the bounds as well visually represent our MERGE-SPLIT works. Here we define four lineage graphs:

(1) **Random Tree (RT)** ($|\mathbb{V}| = 2000$) - Tree generated with random branching factor from $[d_{\min}, d_{\max}]$, where $d_{\min}$ and $d_{\max}$ are the minimum and maximum degrees, respectively.
(2) **Enterprise Data Pipeline (EDP)** ($|\mathbb{V}| = 2000$) - Directed Graph (with some cycles) where edges represent read-write dependencies between compute nodes in a data processing pipeline. There exist cyclic dependencies of length ≤ 2.
(3) **Bidirectional Attention Flow (BIDAF)** ($|\mathbb{V}| = 500$) [42] - The compute execution DAG of BIDAF, which is different from RT or EDP due to the presence of many branching components and single-linked lists.
(4) **Residual Network for Image Recognition (ResNet)** ($|\mathbb{V}| = 2000$) [13] - The compute execution DAG of a random subgraph of ResNet. ResNet is interesting because it contains many short residual connections, which is a feature that isn't present in the other graphs.

For example, for EDP, its outputs represent transformed/cleaned datasets, while the outputs of BIDAF/ResNet are classification tensors. Therefore, during perturbation, *maintaining the existence of random walks between the graphs' inputs and outputs is critical to retaining the semantics of the causal processes they encapsulate.* We

call the set of random walks between arbitrary nodes as the graph's *lineage*, and the foremost facet of MERGE-SPLIT is to preserve lineage as defined in Eq. 1. We empirically verify the bounds in 3 by performing alternating random merges and splits on $|V| = 50$ subgraphs of **RT, EDP, and BIDAF**. The maximum predicted upper bound per operation as well as the actual Δy_{max} is shown in Fig. 3. In each, there exist multiple x_s due to the multiplicity of the $\lambda_i = 0$. Note that red-× represents the total variation distance induced for a particular x_s in the eigenspace of all stationary distributions, while a blue-× represents the same for a split. The black line is the maximum upper bound predicted across all x_s at a particular iteration. In all graphs, this indeed upper bounds the total variation distance in the steady-state distribution, thus minimizing $\vec{\delta}s$ minimizes the change in x_s between G and G'. We conjecture that maintaining x_s also maintains the graph's overall structure with respect to random walks, (see Sec. 5 and Sec. 7).

5 ITERATIVE MERGE-SPLIT

Algorithm 1 Computing Directed RSS and the Potential Function

```
function compute_directed_rss(G: graph)
    directed_rss = [···] HashTable: 𝕆 → δ⃗
    for 𝕄_{i,j} ∈ S_τ^𝕄 do
        directed_rss[𝕄_{i,j}] = ··· Kx1 from Eq. 7
    end for
    for 𝕊_k ∈ S_τ^𝕊 do
        directed_rss[𝕊_k] = ··· Kx1 from Eq. 8
    end for
    return directed_rss
end function

function compute_potential_function(directed_rss, δ⃗(τ))
    φ_τ = ··· potential function
    for 𝕆 ∈ directed_rss do
        δ⃗_𝕆 = directed_rss[𝕆]
        φ_τ(𝕆) = ··· probability from Eq. 11 using (𝕆, δ⃗_𝕆, δ⃗(τ))
    end for
    return φ_τ
end function
```

The previous sections connect the atomic merge and split operations to lineage and measure how each usage affects Directed-RSS. This section considers how to *iteratively* apply each operation to attain a sufficiently perturbed graph without destruction to its overall structure. Rather than choose the operation that has minimal change, apply it and then iterate, We define a *potential function* to select when and where to apply MERGE-SPLIT in order to minimize the *Aggregate* Directed-RSS and verify its performance on our graphs by comparing it to baselines.

Since each operation compounds the aggregate difference in structure and can greatly affect the accumulated spectral similarity, we use a potential function to **minimize** the Aggregate Directed-RSS. After τ operations, let $\vec{\delta}(\tau) \in \mathbb{C}^K$ be the Aggregate Directed-RSS. Let $S_\tau^{\mathbb{M}} = \{\mathbb{M}_{i,j} \cdots\}$ and $S_\tau^{\mathbb{S}} \in \{\mathbb{S}_k \cdots\}$ be the remaining merge and split operations respectively. The probability of selecting

Algorithm 2 Merge Split

```
function merge_split(G: graph)
    directed_rss = compute_directed_rss(G)
    δ⃗(τ) = [···] Kx1 zero vector
    τ = 0
    while length(directed_rss) > 0 do
        φ_τ = compute_potential_function(···)
        𝕆 ~ potential_function
        if 𝕆 = 𝕄_{i,j} then
            G = execute_merge(G, 𝕆)
        else
            G = execute_split(G, 𝕆)
        end if
        δ⃗(τ+1) = δ⃗(τ) + φ_τ(𝕆)
        for edge in edges_incident(G, 𝕆) do
            directed_rss.pop(edge) pop edges incident to O
        end for
        for node in nodes_incident(G, 𝕆) do
            directed_rss.pop(node) pop nodes incident to O
        end for
        directed_rss.pop(𝕆)
        τ += 1
    end while
    return G
end function
```

an operation $\mathbb{O} \in S_\tau^{\mathbb{M}} \cup S_\tau^{\mathbb{S}}$ in terms of the potential function $\phi_\tau()$:

$$\kappa(\mathbb{O}) = \frac{1}{\|\vec{\delta}(\tau) + \vec{\delta}_{\mathbb{O}}\|}$$
$$\phi_\tau(\mathbb{O}) = \frac{\kappa(\mathbb{O})}{\sum_{\mathbb{M}_{u,v} \in S_\tau^{\mathbb{M}}} \kappa(\mathbb{M}_{u,v}) + \sum_{\mathbb{S}_u \in S_\tau^{\mathbb{S}}} \kappa(\mathbb{S}_u)} \tag{11}$$

The update for $\vec{\delta}$ after this operation: $\vec{\delta}(\tau+1) = \vec{\delta}(\tau) + \vec{\delta}_O$. These steps are reflected in Algo. 1 and Algo. 2.

Semantics of Procedure and Runtime: The calculation of Eq. 6 is dependent on each vertex and its immediate neighborhood, so an operation of one vertex, sets a downstream chain reaction of necessary recalculations of $\vec{\delta}_{\mathbb{O}}$, which is computationally expensive. However, by *masking* the community of nodes with edges directly incident to the vertices in the operation, $\vec{\delta}_{\mathbb{O}}$ becomes stationary. Thus, x_n will only be transformed at mutually exclusive indices (or vertices) during subsequent operations avoiding the need to recalculate the Directed-RSS. The calculation of ϕ_τ is dependent on the size of $S_\tau^{\mathbb{M}}$ and $S_\tau^{\mathbb{S}}$. As each set decreases in size by $O(d_{\min})$ each τ, the stopping criterion is when $|S_\tau^{\mathbb{M}}| = |S_\tau^{\mathbb{S}}| = 0$, which takes approximately $O(K|\mathbb{V}|^2/d_{\min})$ (see supplemental). The overall run-time of Iterative MERGE-SPLIT, where β is a hyper-parameter indicating the number of times Directed-RSS is recalculated and $S_{\mathbb{S}}$ and $S_{\mathbb{M}}$ are repopulated, is:

$$O(\beta K((|V| + |E|)d_{\max} + |V|^2/d_{\min})) \tag{12}$$

Measuring the Aggregate Directed-RSS: The ability of our potential function to minimize the Aggregate Directed-RSS is experimentally demonstrated on each of the four sample graphs. Two

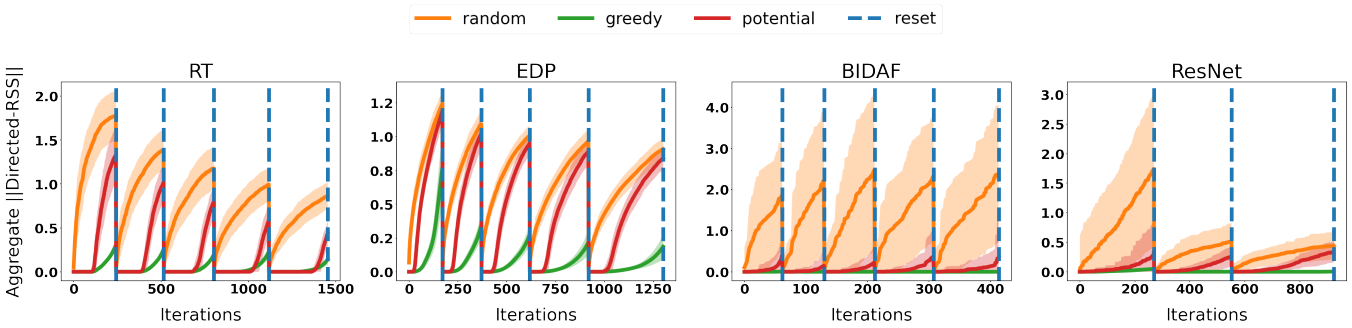

Figure 4: Aggregate Directed-RSS Norm ($||\vec{\delta}(\tau)||$) per the *random*, *greedy*, and *potential* heuristics, for RT, EDP, BIDAF, and ResNet. The results indicate that *random* and *greedy* minimizes $\vec{\delta}(\tau)$ relative to *random*. Shaded regions show per-heuristic variance over 100 trials.

heuristics, in addition to the potential function, are used for comparisons:

(1) *random* - Randomly sample $\mathbb{O}_\tau \in S_{\mathbb{M}} \cup S_{\mathbb{M}}$.
(2) *greedy* - Pick $\mathbb{O}_\tau \in S_{\mathbb{M}} \cup S_{\mathbb{M}}$ such that $\kappa_\tau(\mathbb{O})$ is minimal.
(3) *potential* - Sample $\mathbb{O}_\tau$ from $S_{\mathbb{M}} \cup S_{\mathbb{M}}$ using $\phi_\tau(\mathbb{O})$

The effect of using each heuristic on RT, EDP, BIDAF, and ResNet is measured by Aggregate Directed-RSS Norm ($||\vec{\delta}(\tau)||$) after each application of Merge-Split, averaged over 100 trials (Fig. 4). We use $\beta = 5$ passes for RT, EDP, and BIDAF, and $\beta = 3$ passes for ResNet since the graph sizes are different. For all graphs, we used $K = 200$ eigenvectors to calculate $\vec{\delta}$. The *potential* appears to have sigmoidal growth of $||\vec{\delta}(\tau)||$ for RT, EDP, and ResNet, as evidenced by its exponential growth and decay; however, for BIDAF, this is only a portion of the sigmoid leading up to its inflection (exponential). In general, the $||\vec{\delta}(\tau)||$ of the heuristics is: *random* > *potential* > *greedy*. The leveling-off behavior of *random* and *potential* indicates the distribution of $\vec{\delta}_{\mathbb{S}_k}$ and $\vec{\delta}_{\mathbb{M}_{i,j}}$ is right-skewed. So *greedy* will pick merges and splits that are on the tail-end of this distribution, causing the $||\vec{\delta}(\tau)||$ to grow minimally. Note *greedy* sometimes leads to degenerate cases (Fig. 8) where it strictly increases the $|\mathbb{V}|$ tending to lengthen the roots and leaves (RT/EDP) and the single-linked list (BIDAF/ResNet). *potential* and *random* avoid this issue, although *potential* seems to maintain overall structure while obscuring local subgraphs more effectively. The graphs perturbed by *potential* more closely resemble the overall structure of the original and the subgraphs surrounding the red points are more visually distinct than those of *random*. To better assess the visual differences, we sought additional ablations that quantitatively capture the local and global structural changes according to random walks.

6 LOCAL STRUCTURE OBFUSCATION

The preceding sections demonstrate that using *potential* and *greedy* will minimize the Aggregate Directed RSS, which is our proxy for the change in the graph's overall structure; however, a central goal in the publication of lineage graphs is obscuring local structure to ensure the privacy of subgraphs. There exist three major types of threat models that are addressed in previous literature: node, edge, and subgraph deanonymization. Subgraph Deanonymization is a particularly relevant threat model [4] since it is a super-set of node/edge deanonymization. One such example is the Sybil Subgraph attack, whereby a central curator collects edge information from a set of sources and an attacker gives the curator a recognizable subgraph. The curator might anonymize the nodes/edges, but upon its release, the attacker can perform a subgraph isomorphism to identify a known, recognizable subgraph. Once this happens, the attacker can use additional information to deanonymize surrounding regions of their subgraph and potentially the graph as a whole [12, 31, 33]. Iterative applications of Merge-Split increasingly obscure the structure of local subgraphs, and as argued by the following text, improve the difficulty, or even make it impossible, for an attacker to perform a subgraph isomorphism.

Subgraph Identification - Assume there exists a single instance of a N-sized subgraph, H_N, in G. An attacker must search for a N-sized orbit around each node and some subset of the orbit's edges to verify an isomorphism to H_N, thus taking $O(|\mathbb{V}|(d_{\max})^{N-1})$ [31, 41]. However, how does that complexity change, with a singular application of Merge-Split? Assuming that the attacker knows that Merge-Split was applied once, then H_N will be isomorphic to some subset of valid H_{N-1} or H_{N+1} subgraphs in G'. The number of splits is bound by $V(H_N) = N$ and the number of merges is bound by $E(H_N) = M$. In practice, some merges and splits may create an automorphic set of $N-1$ or $N+1$ subgraphs, so the worst-case complexity is that of searching for N, $N+1$ subgraphs or M, $N-1$ subgraphs: $O(N|\mathbb{V}+1|(d_{\max})^N + M|\mathbb{V}-1|(d_{\max})^{N-2})$.

Extremal analysis of search complexity doesn't directly imply that searching for H_N will be exponential because while the subgraph isomorphism is NP-Complete, there exist fast solvers [35] and techniques to improve search (i.e. searching for multiple subgraphs that have some relation to one another) [20]. However, we *conjecture* that τ applications of Merge-Split should generate a combinatorially large set of potential subgraphs that could correspond to H_N, so even with a fast solver, the search problem is combinatorially large. Also, while we assume in our analysis that the attacker knows τ a-priori, in practice the graph publisher won't release that information so the domain of subgraphs with which H_N could correspond is essentially unbounded. One important caveat is that there exist sequences of perturbations on H_N that spawn

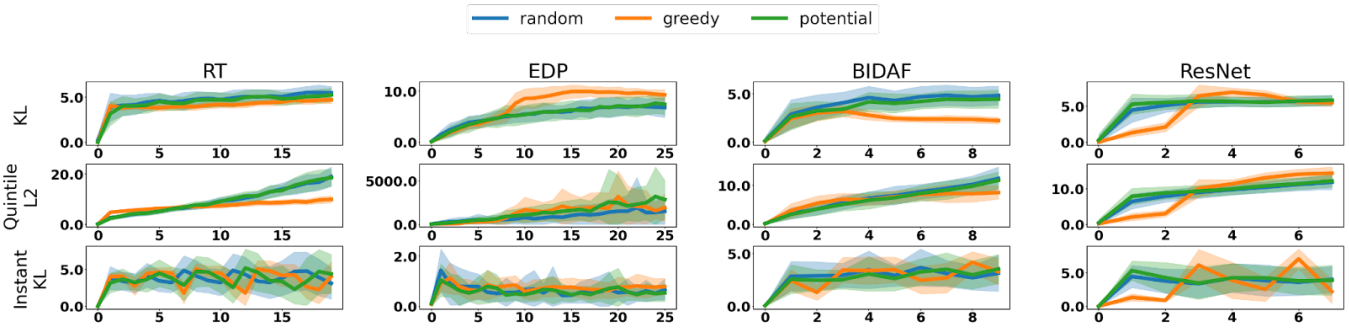

Figure 5: In row-order, the above graphs measure the Overall KL Divergence, Quintile L2 Distance, and Instantaneous KL Divergence. The top row measures the Overall KL Divergence between DGDV(x) and DGDV($C(x)$); the middle row is Quintile L2 Distance between the quintile-filtered graphlets of DGDV(x) and DGDV($C(x)$); the bottom row is the Instantaneous KL Divergence between DGDV(x_{t-1}) and DGDV($C(x_{t-1})$. The x-axis is in units of 50 iterations of Merge-Split. Shaded regions show per-heuristic variance over 100 trials.

subgraphs that are isomorphic to H_N (a simple example is merging and then splitting two nodes in a linked list). We conjecture that the probability of this happening is extremely low because of the randomness within Merge-Split. However, we lack theoretical justification for this claim, so we instead turn to empirical measurement to justify that iterative applications of Merge-Split increasingly spawn subgraphs that differ from the original.

Empirical Privacy Justification - Consider two graphs G and its perturbed counterpart G' after τ applications of Merge-Split. We want the type/frequency of subgraphs within corresponding locations of G and G' to change over time, thus making it more difficult to identify H_N in G'.

To quantify a node's location, we create a vector $\vec{D}_x$ such that $\vec{D}_x[i]$ is the *undirected* shortest path distance between x and the i-th node in a list of the graph's sources and sinks (this list is kept constant between G and G' by restricting Merge-Split from being applied to sources and sinks). A mapping $C(x)$ is the $y \in G'$ that minimizes $||\vec{D}_x - \vec{D}_y||$. We compare the change in Directed Graphlet Degree Vector (DGDV) across all node pairs $(x, C(x))$ where the DGDV follows the definition of Sarajlic et al. [41] for all directed graphlets of size 2, 3, and 4.

Definition 4. ***Directed Graphlet Degree Vector*** *(DGDV) is a "129-dimensional vector encoding the two- to four-node graphlet degrees of the node in the networks; e.g., the i-th coordinate of the DGDV of node n, is the number of times a directed graphlet touches node n at orbit i" [41]. (In practice, we normalize DGDV to represent a probability distribution).*

As seen in Fig. 5, we utilize three measures, Overall KL Divergence, Quintile L2 Distance, and Instantaneous KL Divergence to quantify the magnitude of our perturbations to local structure. The random and potential heuristics increase Overall KL Divergence between G and G' over time. Notice that the Instantaneous KL Divergence between G' at time $t-1$ and t is positive and generally seems to increase over time for random and greedy; however, we notice oscillatory behavior for greedy and slightly oscillatory behavior for random and potential in RT and EDP (that eventually converges). This evidence indicates that iterative applications of the random and potential heuristics tend to increasingly "fuzz" the type and orientation of graphlets at corresponding node locations. Interestingly, we notice that in both BIDAF and ResNet, the greedy heuristic increases the Overall KL Divergence until a certain point, after which it decreases to a converged value. This, combined with the oscillatory behavior of greedy's Instantaneous KL divergence, ties into our visual observations that greedy sometimes results in degenerate cases where it might simply lengthen or shorten the single-linked list components of BIDAF and ResNet as mentioned in Sec. 5.

While the Overall KL Divergence metric measures the changes in DGDV across ALL orientations/types of graphlets an attacker might choose only to focus on searching for infrequently occurring graphlets, as those would be more easily recognizable. Our analysis of the Quintile L2 distance shows that all heuristics tend to iteratively increase the Euclidean distance between the quintile-filtered graphlets. This implies that iterative applications of Merge-Split will obfuscate infrequently occurring types and orientations of graphlets, thus making it more difficult for an attacker who is searching for an infrequently occurring subgraph.

7 LOCAL PERTURBATION AND GLOBAL PRESERVATION

Directed-RSS upper bounds the instantaneous change in the steady-state distribution (Sec 3), but only for eigenvectors of $\lambda_i = 0$; for all other eigenvectors, this bound is irrelevant. Therefore, it must be shown that the minimization of $||\vec{\delta}(\tau)||$ across both the steady-state and non-steady-state eigenvectors achieves the goal of maintaining overall structure with respect to random walks. Additionally, a few degenerate cases from *greedy* (as mentioned in Sec. 5 and Sec. 6), made us question whether τ applications with *potential* or *greedy* truly create a diverse set of subgraphs. Therefore, a new metric, called Random Walk KL-Divergence ($\mathbb{KL}_{rw}$), is introduced that measures the change in probability of traveling from vertex u to vertex v in l-steps in G', where l is the length of the shortest path, at least 1, in G. Notice the minimization of $\mathbb{KL}_{rw}$ is a stronger

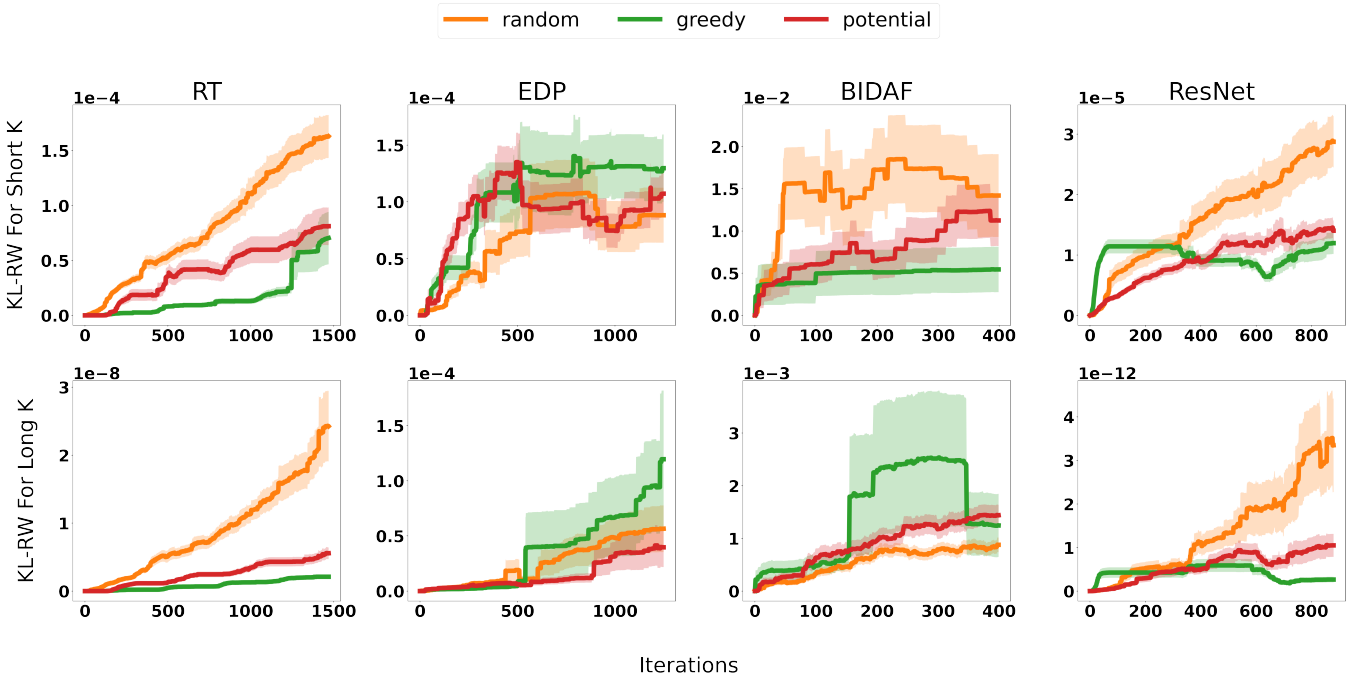

Figure 6: Random-KL Divergence (KL-RW or $\mathbb{KL}_{rw}$) for "short" (first row), and "long" (second row) probes. Results indicate that in 3/4 of graphs, all heuristics disrupt short paths several orders of magnitude more than long paths. Shaded regions show per-heuristic variance over 100 trials.

notion of *lineage*: not only are the paths maintained, but so is the probability of traveling from a path's starting point to its endpoint.

Definition 5. ***Random-Walk KL Divergence ($\mathbb{KL}_{rw}$):*** *Define $\mathbb{KL}_{rw}$ to be the distance-weighted KL-Divergence [15] between the probability of reaching v_i from u_i in l-steps on G versus l' steps on G'. Note that this requires that u and v remain untouched by Merge-Split, so all operations involving these nodes are removed from $S_\tau^{\mathbb{M}}$ and $S_\tau^{\mathbb{S}}$. The probability of reaching v_i from u_i is represented by $p_{u_i,v_i} = (e_{u_i}^T(I - L_{RW})^K)[v_i]$, while on G', $p'_{u_i,v_i} = (e_{u_i}^T(I - L'_{RW})^K)[v_i]$, where e_{u_i} and e_{v_i} are basis vectors.*

$$\mathbb{KL}_{rw} = \frac{1}{\mathcal{B}} \sum_{1 \le i \le \mathcal{B}} |l_i - l'_i| * \mathbb{KL}([p_{u_i,v_i}, 1-p_{u_i,v_i}], [p'_{u_i,v_i}, 1-p'_{u_i,v_i}]) \tag{13}$$

To measure local and global structural change, we sampled $\mathcal{B} = 0.1|\mathbb{V}|$ "short" ($l_i < \mu_l$) and "long" ($l_i > \mu_l$) probes from each graph and re-ran the experiments from Sec. 5. As seen in Fig. 6, for all heuristics, the magnitude of $\mathbb{KL}_{rw}$ for "short" probes is several magnitudes greater than "long" probes across all graphs, except in EDP where *potential* and *random* are only half that of "short" probes. This confirms our assertion that iterative applications of Merge-Split obscure a graph's local structure to a higher degree than its global structure. Additionally, across 3/4 graphs for the "long" probes, *potential* results in a smaller change in $\mathbb{KL}_{rw}$ compared to *random*. This additionally confirms our assertion that the use of our *potential* function is more effective at minimizing overall random walk structural change relative to *random*. *greedy* has unpredictable performance across both "short" and "long" probes, as well as larger variance, which we expand on in the Limitations.

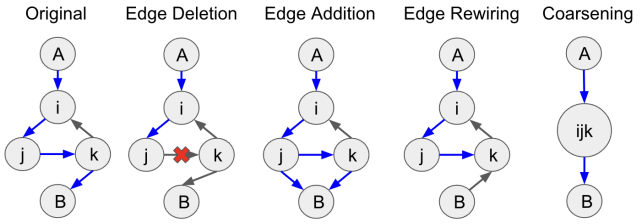

Figure 7: The effect of perturbation/coarsening on paths. Blue edges indicate edges that are on path from $A->B$ on the Original Graph.

8 RELATED WORKS

Publishing graphs that protect certain features has a long research history.

Synthetic graph generation is the process by which graphs are built up non-deterministically following a set of rules and probabilities to match a desired property/distribution. Zhu et. al [47] assigned "mass" to nodes based on their degree in a template graph and used Field Theory to generate undirected edges between the nodes, resulting in degree-based differential privacy guarantees.

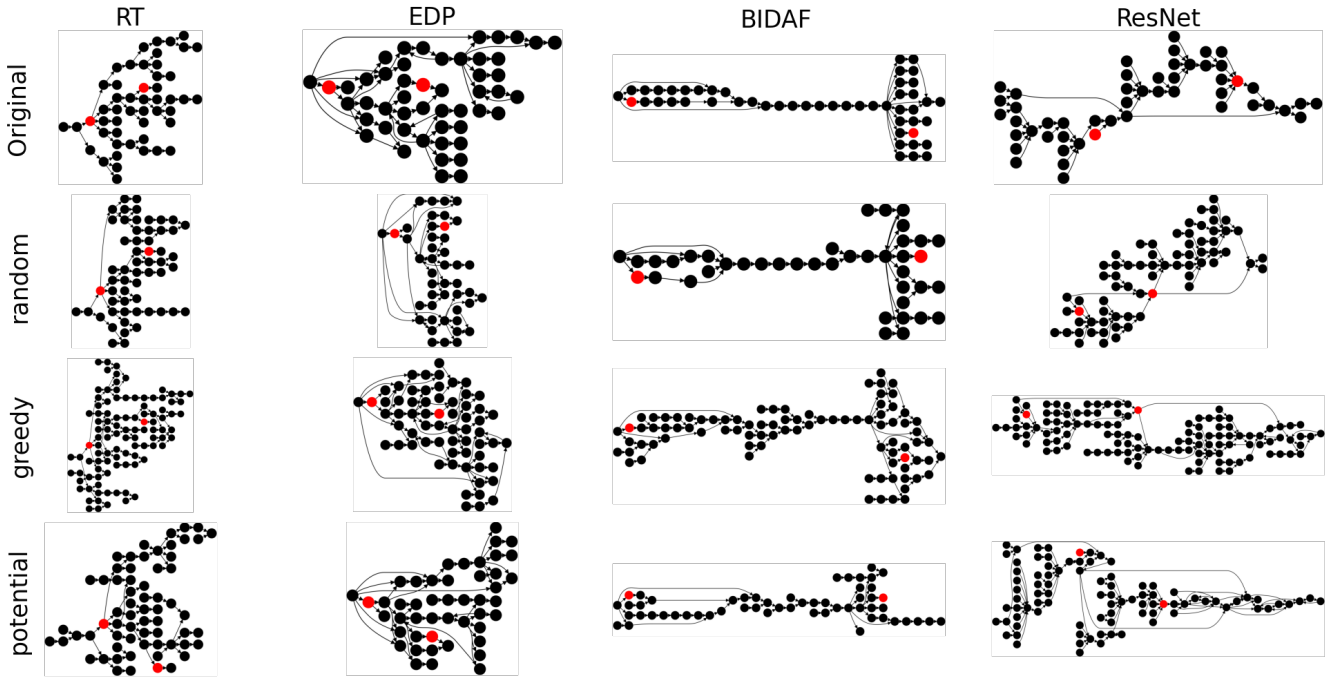

Figure 8: Visualizations of $|\mathbb{V}| = 50$ subgraphs generated by the heuristic set for $\beta = 1$. Red Nodes correspond to fixed points exempt from Merge-Split.

Karwa et. al [40] decomposed a template graph into its degree-unique K-sized subgraphs, perturbed the degrees with Laplacian noise, and rejoined the subgraphs to create a privacy-ensured graph.

Hay et. al [12] considered subgraph isomorphism threat models and proposed randomized edge insertion/deletion to perturb the local structural properties of social networks while retaining their global characteristics, like average path length or diameter. Cheng et. al [4] formalized the protection of subgraphs with heuristic-driven edge creation/deletion by introducing the k-isomorphism privacy goal, which asserts that a node and edge-secure graph should consist of k-copies of particular subgraphs in order to ensure that the probability of finding the right copy is 1/k. In contrast to edge manipulation, Mittal et al. [30] considered the preservation of random walks for social networks by constructing graphs whose edges are t-length random walks of a template social network and showed the relation between t, the mixing time of the graph, and privacy.

Merge-Split builds off these perturbation and threat models they consider, but it uses merges and splits for the nodes/edges of *directed, lineage* graphs, which retain the random walk and spectral properties of the template. Like these works, our formulation considers a central distributor of the networks; however, Liu et al. [22] proposed a local perturbation model in which independent curators inject noise into their local subgraphs before publication to a central entity. We theorize that Merge-Split could be used as the perturbation model in this scheme, but its structural/privacy is a subject of future analysis.

Why isn't previous research applicable? – Synthetic graph generation isn't suitable for lineage preservation because we are concerned with the release of graphs that not only convey the structural content of the original graph but also its identity and function. While synthetic graph generation may spawn graphs that match the form of a ground truth, inherently the function portrayed by the graph is completely different. The conservation of function makes perturbation a more attractive option by virtue of at least starting with the source of truth; however, edge deletion/creation or coarsening will spawn drastically different graphs in the context of lineage. As seen in Fig. 7, consider the original graph where there exists a single path from $A- > i- > j- > k- > B$. In both Edge Deletion and Edge Rewiring, there no longer exists a path from $A- > B$ and thus the lineage of B is destroyed. In Edge Addition or Coarsening, while lineage is maintained from $A- > B$, the paths themselves and the *probability* of moving from $A- > B$ changes, i.e. in edge addition there are two paths instead of 1 and the shortest path is 3 instead of 4 while in coarsening the shortest path length goes from 4 to 2. Additionally, coarsening decreases the size of the graph, but in our case, we want to maintain its size. To the best of our knowledge, Mittal et. al[30] is the only one to address the preservation of random walks in graph publication; however, their method reduces the number and length of paths, which is critical for lineage preservation.

9 DISCUSSION

Merge-Split is a directed graph perturbation algorithm that addresses non-power-law, *lineage* graphs – an increasingly important class that underlies data pipelines and DAG schedulers. There is an opportunity to apply machine learning to these graphs, but they must be distributed in a way that protects the privacy of subgraphs

while maintaining overall structural properties. For directed graphs, this entails the preservation of random walks between pairs of points in the original and perturbed graphs, which we term lineage. Through experimentation and theoretical analysis, we show that Merge-Split, combined with our *potential* function, perturbs local subgraphs while conserving overall structure, such as long-length random walks and the graph's steady state.

Limitations: Although we instill "weights" to the edges of G, Merge-Split is meant for directed, unweighted graphs. Directed Graphs may contain "heavy" edges, which simplifies the search for subgraphs as described in Section 6. Therefore, more careful consideration must be made for threat models where attackers might use weight information in a subgraph isomorphism search. Additionally, a relevant concern was that *greedy* didn't lead to minimal change in $\mathbb{KL}_{rw}$ for "long" probes, and we tie this to the degenerate cases mentioned in Section 3. *greedy* tended to explode singly-linked-list subgraphs or the root/leaves of the graph, which oscillates the distance-weighted component of $\mathbb{KL}_{rw}$, and thus, its $\mathbb{KL}_{rw}$ for long probes isn't always less than *random* or *potential*. This oscillation also explains why the variance of $\mathbb{KL}_{rw}$ is much larger than other *random* and *potential* on a few graphs in Fig. 6. This opens up future discussion for direct optimization of $\mathbb{KL}_{rw}$ with a learning-based method that might mirror reinforcement-learning methods for graph rewiring [25]. Finally, our local obfuscation privacy argument is built off of empirically measuring changes in DGDV, but it lacks theoretical underpinnings. Future work consists of tightly bounding the number of non-automorphic graphs that can be generated from τ applications of Merge-Split.

REFERENCES

[1] Haldun Aytug, Siddhartha Bhattacharyya, Gary J Koehler, and Jane L Snowdon. 1994. A review of machine learning in scheduling. *IEEE Transactions on Engineering Management* 41, 2 (1994), 165–171.

[2] Lars Backstrom, Cynthia Dwork, and Jon Kleinberg. 2007. Wherefore art thou R3579X? Anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th international conference on World Wide Web*. 181–190.

[3] Anna D Broido and Aaron Clauset. 2019. Scale-free networks are rare. *Nature communications* 10, 1 (2019), 1017.

[4] James Cheng, Ada Wai-chee Fu, and Jia Liu. 2010. K-Isomorphism: Privacy Preserving Network Publication against Structural Attacks. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data* (Indianapolis, Indiana, USA) *(SIGMOD '10)*. Association for Computing Machinery, New York, NY, USA, 459–470. https://doi.org/10.1145/1807167.1807218

[5] Wenliang Cheng, Chengyu Wang, Bing Xiao, Weining Qian, and Aoying Zhou. 2016. On statistical characteristics of real-life knowledge graphs. In *Big Data Benchmarks, Performance Optimization, and Emerging Hardware: 6th Workshop, BPOE 2015, Kohala, HI, USA, August 31-September 4, 2015. Revised Selected Papers 6*. Springer, 37–49.

[6] Angel Daruna, Lakshmi Nair, Weiyu Liu, and Sonia Chernova. 2021. Towards robust one-shot task execution using knowledge graph embeddings. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 11118–11124.

[7] Wei-Yen Day, Ninghui Li, and Min Lyu. 2016. Publishing graph degree distribution with node differential privacy. In *Proceedings of the 2016 International Conference on Management of Data*. 123–138.

[8] Cynthia Dwork. 2009. The differential privacy frontier. *Theory of Cryptography Conference* (2009), 496––502.

[9] Apostolos Gerasoulis and Tao Yang. 1992. A comparison of clustering heuristics for scheduling directed acyclic graphs on multiprocessors. *journal of parallel and distributed computing* 16, 4 (1992), 276–291.

[10] Kwang-Il Goh, Eulsik Oh, Hawoong Jeong, Byungnam Kahng, and Doochul Kim. 2002. Classification of scale-free networks. *Proceedings of the National Academy of Sciences* 99, 20 (2002), 12583–12588.

[11] Michael Golosovsky. 2017. Power-law citation distributions are not scale-free. *Physical Review E* 96, 3 (2017), 032306.

[12] Michael Hay, Gerome Miklau, David Jensen, Philipp Weis, and Siddharth Srivastava. 2007. Anonymizing social networks. *Computer science department faculty publication series* (2007), 180.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[14] Jacob Imola, Takao Murakami, and Kamalika Chaudhuri. 2021. Locally Differentially Private Analysis of Graph Statistics.. In *USENIX Security Symposium*. 983–1000.

[15] James M Joyce. 2011. Kullback-leibler divergence. In *International encyclopedia of statistical science*. Springer, 720–722.

[16] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. 2011. Private analysis of graph structure. *Proceedings of the VLDB Endowment* 4, 11 (2011), 1146–1157.

[17] Vishesh Karwa and Aleksandra B Slavković. 2012. Differentially private graphical degree sequences and synthetic graphs. In *Privacy in Statistical Databases: UNESCO Chair in Data Privacy, International Conference, PSD 2012, Palermo, Italy, September 26-28, 2012. Proceedings*. Springer, 273–285.

[18] Shiva Prasad Kasiviswanathan, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. 2013. Analyzing graphs with node differential privacy. In *Theory of Cryptography: 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*. Springer, 457–476.

[19] C-Y Lee, Selwyn Piramuthu, and Y-K Tsai. 1997. Job shop scheduling with a genetic algorithm and machine learning. *International Journal of production research* 35, 4 (1997), 1171–1191.

[20] Yuan Li, Alexander Razborov, and Benjamin Rossman. 2017. On the AC^0 Complexity of Subgraph Isomorphism. *SIAM J. Comput.* 46, 3 (2017), 936–971.

[21] Gipsi Lima-Mendez and Jacques Van Helden. 2009. The powerful law of the power law and other myths in network biology. *Molecular BioSystems* 5, 12 (2009), 1482–1493.

[22] Peng Liu, YuanXin Xu, Quan Jiang, Yuwei Tang, Yameng Guo, Li-e Wang, and Xianxian Li. 2020. Local differential privacy for social network publishing. *Neurocomputing* 391 (2020), 273–279.

[23] Andreas Loukas and Pierre Vandergheynst. 2018. Spectrally approximating large graphs with smaller graphs. In *International Conference on Machine Learning*. PMLR, 3237–3246.

[24] Chengzhi Lu, Wenyan Chen, Kejiang Ye, and Cheng-Zhong Xu. 2020. Understanding the Workload Characteristics in Alibaba: A View from Directed Acyclic Graph Analysis. In *2020 International Conference on High Performance Big Data and Intelligent Systems (HPBD&IS)*. 1–8. https://doi.org/10.1109/HPBDIS49115.2020.9130578

[25] Yao Ma, Suhang Wang, Tyler Derr, Lingfei Wu, and Jiliang Tang. 2019. Attacking graph convolutional networks via rewiring. *arXiv preprint arXiv:1906.03750* (2019).

[26] Ricardo José Machado and Armando Freitas Da Rocha. 1991. The combinatorial neural network: a connectionist model for knowledge based systems. In *Uncertainty in Knowledge Bases: 3rd International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU'90 Paris, France, July 2–6, 1990 Proceedings 3*. Springer, 578–587.

[27] Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, DB Tsai, Manish Amde, Sean Owen, et al. 2016. Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research* 17, 1 (2016), 1235–1241.

[28] Russell Merris. 1998. A note on Laplacian graph eigenvalues. *Linear algebra and its applications* 285, 1-3 (1998), 33–35.

[29] Alan Mislove, Massimiliano Marcon, Krishna P Gummadi, Peter Druschel, and Bobby Bhattacharjee. 2007. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. 29–42.

[30] Prateek Mittal, Charalampos Papamanthou, and Dawn Song. 2012. Preserving link privacy in social network based systems. *arXiv preprint arXiv:1208.6189* (2012).

[31] Arvind Narayanan and Vitaly Shmatikov. 2006. How to break anonymity of the netflix prize dataset. *arXiv preprint cs/0610105* (2006).

[32] Arvind Narayanan and Vitaly Shmatikov. 2009. De-anonymizing social networks. In *2009 30th IEEE symposium on security and privacy*. IEEE, 173–187.

[33] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, et al. 2021. Efficient large-scale language model training on gpu clusters using megatron-lm. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–15.

[34] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2015. A review of relational machine learning for knowledge graphs. *Proc. IEEE* 104, 1 (2015), 11–33.

[35] Miles Ohlrich, Carl Ebeling, Eka Ginting, and Lisa Sather. 1993. Subgemini: Identifying subcircuits using a fast subgraph isomorphism algorithm. In *Proceedings of the 30th International Design Automation Conference*. 31–37.

[36] Romualdo Pastor-Satorras, Claudio Castellano, Piet Van Mieghem, and Alessandro Vespignani. 2015. Epidemic processes in complex networks. *Reviews of modern physics* 87, 3 (2015), 925.

[37] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. (2017).

[38] Marco Piccininni, Stefan Konigorski, Jessica L Rohmann, and Tobias Kurth. 2020. Directed acyclic graphs and causal thinking in clinical risk prediction modeling. *BMC medical research methodology* 20, 1 (2020), 1–9.

[39] Nataša Pržulj. 2007. Biological network comparison using graphlet degree distribution. *Bioinformatics* 23, 2 (2007), e177–e183.

[40] Alessandra Sala, Xiaohan Zhao, Christo Wilson, Haitao Zheng, and Ben Y Zhao. 2011. Sharing graphs using differentially private graph models. In *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. 81–98.

[41] Anida Sarajlić, Noël Malod-Dognin, Ömer Nebil Yaveroğlu, and Nataša Pržulj. 2016. Graphlet-based characterization of directed networks. *Scientific reports* 6, 1 (2016), 35098.

[42] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603* (2016).

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

[44] Daniel A Spielman and Shang-Hua Teng. 2011. Spectral sparsification of graphs. *SIAM J. Comput.* 40, 4 (2011), 981–1025.

[45] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239* (2022).

[46] Matei Zaharia, Reynold S Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J Franklin, et al. 2016. Apache spark: a unified engine for big data processing. *Commun. ACM* 59, 11 (2016), 56–65.

[47] Hong Zhu, Xin Zuo, and Meiyi Xie. 2019. DP-FT: A differential privacy graph generation with field theory for social network data release. *IEEE Access* 7 (2019), 164304–164319.