SCANS: Efficient Densest Subgraph Discovery over Relational Graphs without Materialization

Niu Yudong, Yuchen Li
Singapore Management University
Singapore, Singapore
{ydniu.2018@phdcs,yuchenli@}.smu.edu.sg

Jiaxin Jiang
National University of Singapore
Singapore, Singapore
jxjiang@nus.edu.sg

Laks V.S. Lakshmanan
University of British Columbia
Vancouver, Canada
laks@cs.ubc.ca

ABSTRACT

Driven by applications such as fraud detection and protein complex discovery, there has been extensive work on developing exact or approximation algorithms for efficiently finding densest subgraphs of graphs arising in applications such as financial transactions and protein-protein interactions. At the core of these applications is the application data in the form of relations from which the corresponding graphs, called relational graphs, are derived which are then used to find the densest subgraphs. Most existing approaches assume that these relational graphs are already derived and materialized and focus on efficient discovery of the densest subgraph. Unfortunately, materializing these graphs can be resource-intensive, which thus limits the practical usefulness of existing algorithms over large datasets. To mitigate this, we propose a novel SketChbAsed deNsest Subgraph discovery (SCANS) system. A notable feature of SCANS is that it offers user-defined APIs for customizing peeling algorithms, a popular class of algorithms for densest subgraph discovery, for different density metrics, without the need for graph materialization. Our unique *sketch-based peeling* algorithm forms the core of SCANS. Following the peeling paradigm, it utilizes sketches to efficiently estimate peeling coefficients and subgraph densities at each peeling iteration and thus avoids materializing the relational graph completely. Through extensive experiments, we confirm the efficacy and efficiency of SCANS, reaching orders of magnitude speedups compared to the conventional baselines with materialization while consistently showing accuracy beyond 95%.

PVLDB Reference Format:
Niu Yudong, Yuchen Li, Jiaxin Jiang, and Laks V.S. Lakshmanan. SCANS: Efficient Densest Subgraph Discovery over Relational Graphs without Materialization. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/YudongNiu/SCANS/blob/main/README.md.

1 INTRODUCTION

Discovering dense subgraphs in a given network has broad applications in different domains such as system optimization by social piggybacking [20], discovery of protein complexes [37, 45], information dissemination analysis to discover filter bubbles and echo chambers [31], and forms a building block of many graph problems including reachability queries [13]. Most prior studies on densest subgraph discovery assume that a materialized relational graph is available and accessible with low latency. However, in many real-world scenarios, relational graphs [44] that capture complex relationships between entities should be carefully extracted from heterogeneous data sources, such as knowledge graphs (KGs) [23, 25] and relational databases (RDBMS) [2, 39]. For example, the protein-protein interaction graph needs to be extracted from a biological repository such as RCSB Protein Databank[1] by connecting proteins co-occurring in one or more biological processes.

In this study, we follow the *meta-path* semantics [16, 33, 40, 49] —- a prevalent method for deriving relational graphs from diverse data sources. Our core objective is to address the challenge of *densest subgraph discovery* (DSD) within relational graphs shaped by a specific meta-path $\mathcal{M}$, while avoiding the need to materialize them.

Applications. The discovery of densest subgraphs in relational graphs is pivotal in numerous real-world contexts, including:

- *Fraud Detection.* In e-commerce platforms, fraud, such as promotion abuse, is common where fraudsters exploit promotions by creating fake accounts to unfairly benefit from offers. They often log into various accounts using a limited number of devices. This relationship between accounts and devices can be represented by meta-paths "(user)→(*account*) → (*device*)". Identifying densest subgraphs in this context serves as a strong indicator of potential fraudulent activities, aiding in the detection and prevention of such schemes [10].
- *Identifying Filter Bubbles.* Filter bubbles and echo chambers occur in social networks, creating environments where users are exposed primarily to ideas and viewpoints similar to their own, thus limiting content diversity and exposure [31]. These networks, structured through meta-paths "(*user*) → (*group*)", enable the mapping of user-group interactions. Identifying the densest subgraphs within these mappings is crucial, as it highlights the concentration of similar viewpoints, signifying the existence of filter bubbles. This process is essential for addressing information diversity challenges and mitigating echo chamber effects in social media platforms.
- *Protein Complex Discovery.* Protein-protein interaction (PPI) networks offer a detailed map of the intricate interactions among proteins that are fundamental to physiological processes. These networks are adeptly characterized through meta-paths of the form "(*protein*) → (*function*)", which articulate the functional pathways connecting proteins. Densest subgraphs within PPI networks

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

[1] https://www.rcsb.org/

have been found to correspond to protein complexes [37, 45]. Such complexes are not only vital for understanding the specific roles and mechanisms of proteins in cellular activities but also for unveiling potential targets for therapeutic intervention.

Challenges. Most approaches for the DSD problem assume that the underlying relational graph is extracted and materialized. This requires enumerating all instances of $\mathcal{M}$, which poses two significant challenges. First, the exhaustive enumeration and subsequent materialization of the relational graph are inherently resource-intensive, requiring substantial time and memory, especially with large-scale data sources. For example, it takes more than 1200 seconds to materialize the co-purchase graph from a TPCH benchmark [46]. This complexity is exacerbated by the need for complex operations such as multiple joins in RDBMS or traversing meta-path instances in KGs. Second, the vast diversity of meta-paths in heterogeneous data sources further complicates this approach. Considering the potentially thousands of meta-paths, materializing a relational graph for each meta-path can become prohibitively expensive.

Our Contributions. To address these challenges, we propose a novel system, SketCh-bAsed deNsest Subgraph discovery (SCANS) for DSD. Our system distinguishes itself from prior studies by introducing the following novel optimizations.

Sketch-Peeling Algorithm. Sketching, an efficient technique primarily recognized for estimating distinct value counts [6], is repurposed in SCANS to serve a novel role in relational graph analysis for DSD. This use of sketching remains largely unexplored in existing studies. Specifically, by integrating sketching with the peeling algorithm, SCANS *directly estimates* peeling coefficients and densities from heterogeneous data sources. This strategic approach sidesteps the extensive resource demands typically associated with the materialization of relational graphs. Our system showcases superior time and space efficiency over contemporary methods for detecting the densest subgraph which rely on materialized relational graphs.

Lazy Sketch Maintenance. SCANS leverages the well-known KMV sketches [4] and introduces a *lazy sketch maintenance* strategy to update KMV sketches efficiently during the dense subgraph discovery process. Rather than reconstructing KMV sketches at each iteration—a highly time-consuming task—we opt for a selective update mechanism. This approach maintains the sketches' relevance by adjusting them as nodes are removed, with occasional reconstructions based on a set threshold to control the balance between efficiency and the accuracy of density estimations.

Specifically, we make the following contributions.

- We propose SCANS, a novel system that allows data engineers to deploy user-defined, efficient and scalable peeling algorithms for densest subgraph discovery over relational graphs (conceptually) extracted from large heterogeneous data sources (Sec. 3.1).
- We devise the *sketch-based peeling* algorithm, which efficiently constructs and maintains the sketches for peeling coefficients and subgraph density estimation across peeling iterations (Sec. 3.2).
- We deploy our SCANS system for DSD for edge density and theoretically establish that SCANS can achieve $(2+\epsilon)$-approximation efficiently. For triangle-density metric, we devise novel unbiased estimators for the corresponding peeling coefficient and triangle-density based on the sketches (Sec. 4).
- Extensive experiments on real-world datasets and the TPCH benchmark demonstrate that the SCANS system scales well to large relational graphs where baselines fail to terminate in a reasonable time and achieves orders of magnitude speedup in most cases over baselines, while yielding subgraphs with density at least 95% of the optimal density and comparable size (Sec. 5).

2 PRELIMINARIES

In this section, we introduce the basic notations, formulate the problem of finding densest subgraph over relational graphs (DSD), and then review the peeling paradigm for densest subgraph discovery.

2.1 Notations and Problem Formulation

We model a heterogeneous data source as a knowledge graph (KG). Specifically, a KG is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L})$, where $\mathcal{V}$ and $\mathcal{E}$ represent the sets of node and edge instances. An edge instance $e \in \mathcal{E}$ connects two node instances $u, v \in \mathcal{V}$. The function $\mathcal{L}$ maps each node or edge instance, v or e, to its type, $\mathcal{L}(v)$ or $\mathcal{L}(e)$. *Note that our formulation and methods are orthogonal to the format of the heterogeneous data and can be extended to support other data sources such as RDBMS.* For concreteness of exposition, we model a heterogeneous data source as a KG.

Meta-paths, originally introduced in [40], are commonly used to extract relational graphs from KGs [16, 30, 36, 49, 51].

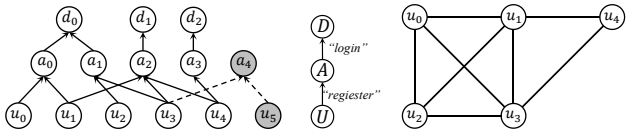
Definition 1 (Meta-path). *An L-hop meta-path is a sequence of node and edge types denoted as $\mathcal{M}(x_0, y_0, x_1, \ldots, x_{L-1}, y_{L-1}, x_L)$, where x_i is the type of the i-th node and y_i is the type of the i-th edge.*

Definition 2 (Matching Instance). *A matching instance M of a meta-path $\mathcal{M}$, denoted as $M \triangleright \mathcal{M}$, is a sequence of node and edge instances $M(v_0, e_0, v_1, \ldots, v_{L-1}, e_{L-1}, v_L)$ in $\mathcal{G}$ satisfying:*

(1) $\forall i \in \{0, \ldots, L\}$, $\mathcal{L}(v_i) = x_i$.
(2) $\forall i \in \{0, \ldots, L-1\}$, $e_i = (v_i, v_{i+1}) \in \mathcal{E}$ *and* $\mathcal{L}(e_i) = y_i$.

When the context is clear, we use $M(v_0, v_1, \ldots, v_L)$ and M interchangeably to denote a matching instance of meta-path $\mathcal{M}$ and use $M(x_i)$ to denote the node v_i in instance M, of the node type x_i.

Definition 3 (Relational Graph). *For a given KG $\mathcal{G}$, a meta-path $\mathcal{M}$ induces a relational graph $H_{\mathcal{M}} = (V_{\mathcal{M}}, E_{\mathcal{M}})$ from $\mathcal{G}$ with node set $V_{\mathcal{M}}$ containing all the nodes in $\mathcal{V}$ that match x_0 in any instance of $\mathcal{M}$ and edge set $E_{\mathcal{M}}$ containing each edge $e = (u, v)$ for any two nodes $u, v \in V_{\mathcal{M}}$ such that there are $M, M' \triangleright \mathcal{M}$ in $\mathcal{G}$ with (1) $M(x_0) = u$, (2) $M'(x_0) = v$, and (3) $M(x_L) = M'(x_L)$.*



(a) Exemplary KG $\mathcal{G}$ (b) Meta-path $\mathcal{M}$ (c) Relational graph H

Figure 1: Exemplary KG $\mathcal{G}$ and relational graph H derived from $\mathcal{G}$ using meta-path $\mathcal{M}(U,$ 'register', $A,$ 'login', $D)$. The unshaded nodes and solid edges in (a) denote the matching graph of $\mathcal{M}$.

Example 1. *Fig. 1 presents an exemplary e-commerce KG with three node types U ("user"), A ("account"), D ("device"). Sequence (U, "register", A, "login", D) denotes a meta-path $\mathcal{M}$, which induces the relational graph H in Fig. 1(c). For instance, two nodes u_0 and u_2 are neighbors in $H_{\mathcal{M}}$ because there exists two instances $M = (u_0, a_0, d_0)$ and $M' = (u_2, a_1, d_0)$ of $\mathcal{M}$. More intuitively, two authors u_0 and u_2 are connected since they ever login through the same device d_0.*

We denote the set of neighbors of u in $H_{\mathcal{M}}$ as $\mathcal{N}(u)$ and the *degree* of u in $H_{\mathcal{M}}$ as $N(u) = |\mathcal{N}(u)|$. Furthermore, $\Lambda = \max_{u \in H} N(u)$ denotes the maximum degree in $H_{\mathcal{M}}$. When the context is clear, we drop the subscript $\mathcal{M}$ and use $H = (V, E)$ to represent a relational graph to simplify the presentation.

Induced Subgraph. Let $H = (V, E)$ be a relational graph and $S \subseteq V$. The induced subgraph of H *w.r.t.* S is the subgraph $H[S] = (S, E[S])$, where $E[S] = \{(u, v) \in E \mid u \in S \wedge v \in S\}$. In the rest of the paper, we abuse the notation S to also denote the subgraph it induced.

Density Metrics $\rho(\cdot)$. We define a density metric ρ for a subset S of vertices as $\rho(S) = \frac{f(S)}{|S|}$, where $f(S)$ represents the weight of the induced subgraph $H[S]$. Our proposed system supports a variety of density metrics, among which $f(S)$ represents the number of edges [21], number of triangles [42] or more generic functions wrt. node degrees [43] within S.

We then formally define the problem of *densest subgraph discovery* (DSD) over relational graphs.

Problem 1 (DSD). *Given a relational graph H and a density metric $\rho(\cdot)$, find a set of vertices $S^* \subseteq V$ such that $\rho(S^*)$ is maximized i.e.,*

$$S^* = \arg\max_{S \subseteq V} \rho(S)$$

For example, in Fig. 1(c), the induced subgraph of the node set $S^* = \{u_0, u_1, u_2, u_3\}$ forms the densest subgraph over the relational graph H in terms of edge-density metric.

In this work, we focus on discovering the set of nodes S^* forming the densest subgraph instead of returning the induced subgraph $H[S^*]$. Nevertheless, after revealing the set of nodes S^*, we can choose to materialize $H[S^*]$ subject to application requirements more efficiently by disregarding nodes outside S^*.

2.2 Peeling Paradigm

Peeling Coefficient $\varphi_S(u)$. The peeling coefficient function, denoted by $\varphi_S(u)$, quantifies a node's contribution to the density of the induced subgraph $H[S]$. Its calculation is dependent on the chosen density metric. We define $\varphi_S(u)$ as follows:

$$\varphi_S(u) = f(S) - f(S \setminus \{u\}) \tag{1}$$

When the context is clear, we drop S and denote the peeling coefficient as $\varphi(u)$ for ease of presentation.

Peeling Paradigm (Algorithm 1). The peeling paradigm is a general iterative approach for the densest subgraph discovery. Given a relational graph $H = (V, E)$, it begins by computing the initial peeling coefficient of each node in V (Line 2) and then obtains a sequence of subgraphs by peeling vertices iteratively (Line 3–6). In each iteration, it peels the node with minimum *peeling coefficient* (Line 4), updates the peeling coefficient for the set of nodes whose peeling coefficient is influenced by the peeling (Line 5), and maintains the subgraph with largest density (Line 6). Finally, the subgraph of the largest density is returned (Line 7). The *peeling coefficient* is determined by the density metric to achieve good approximations of the optimal.

Algorithm 1: `Peeling Paradigm`

Input: Relational Graph $H = (V, E)$
Output: $S \subseteq V$, such that $\rho(S)$ is maxmized

1 $S \leftarrow \emptyset, S' \leftarrow V$;
2 compute the initial *peeling coefficient* $\varphi_{S'}(u)$ for each $u \in S'$;
3 **while** $S' \neq \emptyset$ **do**
4 $\quad v \leftarrow \arg\min_{u \in S'} \varphi_{S'}(u)$ and peel v from S';
5 $\quad$ update $\varphi_{S'}(u)$ for each node u influenced by the peeling;
6 $\quad$ **if** $\rho(S') > \rho(S)$ **then** $S \leftarrow S'$;
7 return S;

The time overhead of the peeling paradigm mainly comes from two parts. The first part is initializing the peeling coefficient of each node $u \in V$. The second part is updating the peeling coefficient for nodes that are influenced by the peeling of the node with smallest peeling coefficient in each iteration. Although the peeling algorithm runs in linear number of iterations, it can still be time-consuming for certain density metrics. For example, the triangle-density [42] requires computing the number of triangles containing each node in $O(n^{2.376})$ times, and the number of nodes influenced by peeling at each peeling iteration can be significant for distance-generalized metrics [7]. As discussed in Sec. 3.2, our proposal can not only avoid the materialization stage but also accelerate the peeling iterations by utilizing sketches for efficiently estimating and updating the peeling coefficients.

3 THE SCANS SYSTEM

In this section, we first present an overview of our proposed SCANS system and APIs. Subsequently, we describe in detail the sketch-peeling algorithm, which constructs and maintains the sketches utilized in the SCANS system.

3.1 Overview of SCANS and APIs

Motivated by the observation that the peeling paradigm for densest subgraph discovery only requires *peeling coefficients* of each node for node removal and the densities of the sequence of generated subgraphs to determine the output subgraph, the SCANS system follows the peeling paradigm and avoids the materialization of the relational graphs by utilizing persistent sketches to efficiently estimate the peeling coefficient and subgraph densities during the peeling iterations.

Crafting the complete SCANS system for different density metrics can be cumbersome for data engineers, given the significant efforts required to construct and maintain the sketches within the system. SCANS streamlines this endeavor and provides a set of interfaces to efficiently implement peeling algorithms for different density metrics. This allows data engineers to concentrate on selecting the best density metric and the corresponding peeling coefficient for their particular real-world application.

```
class SCANS{
    virtual double coefficient(int v, sketches K);
    virtual double density(vector<int> S, sketches K);
    double val(sketch k);
}
```

Listing 1: APIs of SCANS

Application Programming Interfaces (APIs). Listing 1 presents the key APIs in SCANS system through which users can deploy the SCANS system to optimize various density metrics.

- `coefficient()` estimates the *peeling coefficient* for node `v` utilizing the sketches `K`. This API is called when computing the initial peeling coefficients as well as updating the peeling coefficient of influenced nodes after each removal during peeling iterations.
- `density()` estimates the density of the given subgraph `S` utilizing the sketches of nodes in `S` according to density metrics implemented by the user. This API is called to estimate the density of the sequence of subgraphs generated during peeling iterations.
- `val()` gives the estimation of the cardinality of the collection represented by the sketch `k`. This API can be used by users for implementing the APIs `coefficient()` and `density()`.

3.2 Implementation of SCANS

We first review the KMV sketch which is used in SCANS. It was initially proposed for distinct value estimation [4, 6].

Definition 4 (KMV Sketch). *Given a collection $C = \{o_0, o_1, ..., o_{|C|}\}$ of items and an integer $k > 0$, the KMV (k-th Minimum Value) sketch $\mathcal{K}(C)$ is built by independently drawing a number uniformly at random from $(0, 1)$ for each item in C and maintaining the k smallest random numbers. The set of random numbers generated for each item in C is called the* basis *for the KMV sketch. The cardinality of C is estimated as $|\widetilde{C}| = \frac{k}{\mathbb{E}[\zeta]} - 1$, where ζ is a random variable corresponding to the largest number in $\mathcal{K}(C)$ and $\mathbb{E}[\zeta]$ is the expectation of ζ.*

We adopt the KMV sketch in SCANS since it has the following desirable properties:

- *Efficiency:* The KMV sketches can be constructed for each node in the relational graph efficiently from external KGs.
- *Persistency:* The KMV sketches can be maintained persistently during peeling iterations efficiently.
- *Flexibility:* The KMV sketches are compatible with multiple set operations [6] so that it can support estimation of complex peeling coefficients and subgraph densities.

Sketch-Peeling Algorithm. Next, we present the sketch-peeling algorithm as the core of SCANS that constructs and maintains the KMV sketches for the neighborhood of each node in the relational graph efficiently without materialization and utilizes the sketches for the densest subgraph discovery.

Sketch Construction. The naïve approach is to first obtain the neighborhood of each node in H and then construct the KMV sketches as described in Def. 4. In particular, the neighborhood of each node in H can be obtained by propagating nodes in the *matching graph* defined as the follows:

Definition 5 (matching graph). *Given a KG $\mathcal{G}$ and a meta-path $\mathcal{M}$, the* matching graph *of $\mathcal{M}$ over $\mathcal{G}$ is a* multi-level *graph $\mathcal{G}^*(\mathcal{V}^*, \mathcal{E}^*)$ with a node set $\mathcal{V}^* = \bigcup_{0 \le i \le L} \mathcal{V}_i$ and an edge set $\mathcal{E}^* = \bigcup_{0 \le i < L} \mathcal{E}_i$, where $\mathcal{V}_i$ is the set of all node instances of type x_i contained in any matching instance of $\mathcal{M}$ and $\mathcal{E}_i$ consists of all edges of type y_i that connects two nodes between $\mathcal{V}_i$ and $\mathcal{V}_{i+1}$.*

Definition 6 (Successors & Predecessors). *For each node $u \in \mathcal{V}_i$, we use $\mathcal{V}^+(u)$ to denote the nodes in $\mathcal{V}_{i+1}$ connected with u through edge type y_i, i.e., the* successors *of u in $\mathcal{G}^*$; and use $\mathcal{V}^-(u)$ to denote the nodes in $\mathcal{V}_{i-1}$ connected with u through edge type y_{i-1}, i.e., the* predecessors *of u in $\mathcal{G}^*$.*

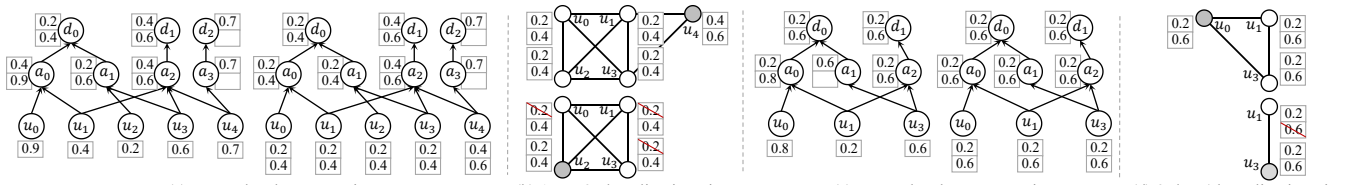
By propagating all nodes in $\mathcal{V}_0$, which equals to node set V of relational graph H, along the matching graph from $\mathcal{V}_0$ to $\mathcal{V}_L$ (*forward propagation*) and then from $\mathcal{V}_L$ to $\mathcal{V}_0$ (*backward propagation*), each node in $\mathcal{V}_0$ will receive all its neighbors. However, obtaining the whole neighborhood of each node can lead to a prohibitive time complexity of $O(|V| \cdot |\mathcal{E}^*|)$ since each node in $\mathcal{V}^*$ potentially retains all the nodes it has received and then propagates these nodes to its successors (during forward propagation) or predecessors (during backward propagation) through edges in $\mathcal{G}^*$.

Fortunately, this prohibitive complexity can be circumvented by utilizing the fact that the KMV sketch construction does not require all neighbors of each node. Instead, it only requires a subset of a node's neighbors corresponding to the smallest k random numbers in the KMV sketch basis. Thus, each node in $\mathcal{V}^*$ only needs to preserve and propagate at most k nodes received during the propagation process.

Specifically, at the beginning, we generate a random number for each node $v \in V$, denoted $r(v)$, as the basis of the KMV sketch. Each node in $\mathcal{V}_{i-1}$ propagates the random numbers it maintains to its successors. Each node in $\mathcal{V}_i$ will retain the k-smallest random numbers it receives. When each node in $\mathcal{V}_L$ has received its random numbers, they will propagate the random numbers back to nodes in $\mathcal{V}_0$, still retaining the k-smallest random numbers at each node in $\mathcal{V}^*$. Finally, each node in $\mathcal{V}_0$ will receive k random numbers forming the KMV sketch for its neighborhood. This sketch construction process is presented as Function `construct()` in Alg. 2, where the operator $\oplus$ extracts the k smallest random numbers from the union of random number collections.

Lazy Sketch Maintenance. We can use the KMV sketches as constructed above to estimate the peeling coefficient $\varphi(v)$ for each node $v \in V$ and the current density $\rho(H)$. However, during the peeling iterations, nodes are iteratively removed and the remaining subgraph is changed continuously. Thus, the KMV sketches can become outdated. A naïve solution is to reconstruct the KMV sketches at each peeling iteration. However, this approach is expensive since it requires reconstructing the KMV sketches $O(V)$ times.

To this end, we propose a *lazy sketch maintenance* approach for efficient maintenance of KMV sketches during peeling iterations. At each peeling iteration, when a node v is removed, we can easily maintain the KMV sketches for nodes in the remaining subgraph by removing the random number $r(v)$ corresponding to v from the KMV sketches. Specifically, for each KMV sketch of size k' containing $r(v)$, we will obtain a KMV sketch of size $k' - 1$ without $r(v)$ after the peeling iteration. Since the original KMV sketch before maintenance retains the k' smallest random numbers



(a) KMV sketch construction (b) 1st & 2nd peeling iterations (c) KMV sketch reconstruction (d) 3rd & 4th peeling iterations

Figure 2: Sketch-peeling algorithm with $k = 2$ and $k^- = 1$. Shaded nodes in subfigures (b) and (d) denote the nodes removed in the peeling iteration and the elements in KMV sketches with red strikethrough are the random numbers corresponding to the removed nodes. Note that subgraphs in (b) and (d) are only for demonstration and not materialized in our algorithm.

corresponding to nodes in the neighborhood, after removing the random number $r(v)$ from the KMV sketch, it still retains the $k' - 1$ smallest random numbers corresponding to nodes in the remaining neighborhood. However, since the accuracy of the KMV sketch is proportional to its size, the *lazy sketch maintenance* can potentially deteriorate the effectiveness of the KMV sketches. To alleviate this problem, we reconstruct the KMV sketches whenever we find they not effective enough for estimation. More precisely, we set a threshold k^- and reconstruct the KMV sketches whenever their size falls below k^-. This lazy sketch maintenance is presented as Function `update()` in Alg. 2.

Sketch-based Peeling. Alg. 2 gives the pseudocode for the sketch-peeling algorithm, which implements the peeling paradigm over relational graphs efficiently based on KMV sketches. It receives a KG $\mathcal{G}$, a meta-path $\mathcal{M}$, the number of sketches $\theta \in \mathbb{Z}^+$ and the sketch size bounds $k, k^- \in \mathbb{Z}^+$ as input, and returns a subset S of V maximizing the density of $H[S]$ for the DSD problem. The algorithm takes the average of estimations obtained through θ KMV sketches for each node in H to achieve good approximation guarantees for peeling coefficients and subgraph densities. It first initializes the empty buffer $\mathcal{K}$ for storing the KMV sketches and array I indicating the index of sketches to be constructed and constructs the θ sketches (Lines 1–2). Then, the peeling iterations proceed until only one node remains (Lines 3–9). In each peeling iteration, the algorithm selects and removes the node with the smallest peeling coefficient by calling API `coefficient` (Lines 4–5). Then, the algorithm calls procedure `update` to perform lazy maintenance of the KMV sketches and obtains the indices I of sketches that need to be reconstructed (Line 6). Then it reconstructs the KMV sketches indicated in I over the current remaining subset S' (Line 7). At the end of each iteration, API `density` is called to estimate the density of $H[S']$ (Line 8) and the subset induces larger density is maintained in S (Line 9). Finally, the subset S induces largest density obtained during the peeling iterations is returned (Line 10).

Example 2. *Fig. 2 illustrates the sketch-peeling algorithm finding the subset induces densest subgraph based on edge-density metric over the relational graph in Fig. 1(c) with sketch size bounds $k = 2$ and $k^- = 1$. First, the KMV sketches are constructed by propagating the random numbers forward (Fig. 2(a) left) and backward (Fig. 2(b) right) along the matching graph. Fig. 2(b) denotes the first and second peeling iteration, where node u_4 and u_2 with peeling coefficient $\varphi(u_4) = \frac{2}{0.6} - 1 = 2.33$ and $\varphi(u_2) = \frac{2}{0.4} - 1 = 4$ are removed respectively. The subsets $S_1 = \{u_0, u_1, u_2, u_3\}$ and $S_2 = \{u_0, u_1, u_3\}$ with estimated density $\rho(S_1) = \frac{(\frac{2}{0.4}-1)\cdot 4}{2\cdot 4} = 2$ and $\rho(S_2) = \frac{(\frac{1}{0.4}-1)\cdot 3}{2\cdot 3} = 0.75$ are generated in the peeling iterations. The KMV sketches are reconstructed as depicted in Fig. 2(c). The third and fourth peeling iterations are presented in Fig. 2(d), where node u_0 and u_3 with peeling coefficient $\varphi(u_0) = \frac{2}{0.6} - 1 = 2.33$ and $\varphi(u_3) = \frac{2}{0.6} - 1 = 2.33$ are removed, resulting in subset $S_3 = \{u_1, u_3\}$ with density $\rho(S_3) = \frac{(\frac{2}{0.6}-1)\cdot 2}{2\cdot 2} = 1.167$. Thus, subset S_1 induces the largest density and is returned as the result.*

Algorithm 2: Sketch-based Peeling

Input: KG $\mathcal{G}$, meta-path $\mathcal{M}$, number of sketches θ, KMV sketch size bounds k, k^-
Output: A set of nodes $S \subseteq V$

1 $S \leftarrow \emptyset, \rho \leftarrow 0, S' \leftarrow V, I \leftarrow \{1, 2, 3, ..., \theta\}$ and $\mathcal{K}[\,][\,] \leftarrow \emptyset$;
2 `construct`($\mathcal{K}, V, I$);
3 **while** $|S'| > 1$ **do**
4 $v \leftarrow \arg\min_{u \in S'}$ `coefficient`$(u, \mathcal{K})$;
5 remove v from S';
6 $I \leftarrow$`update`$(\mathcal{K}, v)$;
7 `construct`$(\mathcal{K}, S', I)$;
8 $\rho' \leftarrow$`density`$(S', \mathcal{K})$;
9 **if** $\rho' > \rho$ **then** $S \leftarrow S', \rho \leftarrow \rho'$;
10 **return** S;
11 **Function** `construct`$(\mathcal{K}, S, I)$**:**
12 **for** $\tau \in I$ **do**
13 **if** $u \in S$ **do** $\mathcal{K}[u][\tau] \leftarrow$ `Rand`$(0, 1)$;
14 **for** $i = 1$ **to** L **do**
15 **for** $u \in \mathcal{V}_i$ **do** $\mathcal{K}[u][\tau] \leftarrow \oplus_{v \in \mathcal{V}^-(u)} \mathcal{K}[v][\tau]$;
16 **for** $i = L - 1$ **to** 0 **do**
17 **for** $u \in \mathcal{V}_i$ **do** $\mathcal{K}[u][\tau] \leftarrow \oplus_{v \in \mathcal{V}^+(u)} \mathcal{K}[v][\tau]$;
18 **Function** `update`$(\mathcal{K}, u)$**:**
19 $I \leftarrow \emptyset$;
20 **for** $\tau = 1$ to θ **do**
21 **for** $v \in S'$ **do**
22 remove $r_\tau(v)$ from $\mathcal{K}[v][\tau]$;
23 **if** $|\mathcal{K}[v][\tau]| < k^-$ **then** $I \leftarrow I \cup \{\tau\}$;
24 **return** I;

Complexity analysis. Next, we give the time and space complexity analysis of the sketch-peeling algorithm.

Theorem 1. *The time complexity of Alg. 2 is $O(k(\theta + \theta^*)(|\mathcal{E}^*| + |V|f(k) + |V|\log|V|))$, where $f(k)$ denotes the time complexity for peeling coefficient estimation using KMV sketch and θ^* denotes the total number of sketches reconstructed during the peeling iterations.*

Proof. There are mainly two contributors to the time complexity of the sketch-peeling algorithm. First, the algorithm needs to construct the KMV sketches. The construction of KMV sketches requires propagating at most k random numbers over the matching graph with $|\mathcal{E}^*|$ edges, which takes $O(k \cdot |\mathcal{E}^*|)$ time. The algorithm constructs θ KMV sketches at the beginning of the algorithm and reconstructs θ^* sketches that are disabled by node removals during peeling iterations. Thus, the time complexity of sketch construction is $O((\theta + \theta^*)k \cdot |\mathcal{E}^*|)$. Then, at each peeling iteration, the algorithm updates the KMV sketches by removing from $\mathcal{K}$ the random number corresponding to the removed node, re-estimates the peeling coefficients with updated KMV sketches, and maintains the min-heap for identifying the node with smallest peeling coefficient. Since $(\theta + \theta^*)$ KMV sketches are constructed for each node, at most $O(k|V|(\theta + \theta^*))$ KMV sketch updates will be performed. For each KMV sketch update, it will takes $f(k)$ time to re-estimate the peeling coefficient and $O(\log |V|)$ time to maintain the min-heap. The KMV sketch update itself can be performed in $O(1)$ utilizing an inverted list from random numbers to KMV sketches. Thus, the time complexity is $O(k|V|(\theta+\theta^*)(f(k)+\log|V|))$. Therefore, the total time complexity of the sketch-peeling algorithm is $O(k(\theta + \theta^*)(|\mathcal{E}^*| + |V|f(k) + |V| \log |V|))$. □

Theorem 2. *The space complexity of Alg. 2 is* $O(k\theta|\mathcal{V}^*| + |\mathcal{E}^*|)$.

Proof. The firsts term $O(k\theta|\mathcal{V}^*|)$ is the memory consumption of KMV sketches and the second term $O(|\mathcal{E}^*|)$ is the memory consumption of the matching graph $\mathcal{G}^*$. □

The time complexity of Alg. 2 is significantly influenced by the number of sketches θ^* that need to be reconstructed due to node removals during peeling iterations. In the worst case, we have $\theta^* = O(V/k)$. Once k nodes are removed, there might exist a node whose KMV sketch becomes invalid since the random numbers in the KMV sketch correspond exactly to the removed k nodes and those numbers are removed from the KMV sketch. However, in the following, we show that θ^* is much smaller in practice than in the worst case especially when the SCANS system is deployed for optimizing edge-density or triangle-density metrics.

4 DEPLOYMENT OF SCANS SYSTEM

In this section, we deploy the SCANS system to four common density metrics to demonstrate the effectiveness and flexibility of SCANS. For edge-density and triangle-density, we give the implementation of APIs for peeling coefficient and subgraph density estimation with theoretical analysis of the sketch-based algorithm. For p-mean density and f-density which generalize the common edge-density, we discuss about the implementation of APIs for these two metrics in our SCANS system.

4.1 Edge-density $\rho_e(S)$

We first deploy the SCANS system over edge-density metric, first proposed by Goldberg [21]:

Definition 7 (Edge-density). *Given a graph* $S = (V_S, E_S)$, *its edge-density is defined as* $\rho_e(S) = \frac{|E_S|}{|V_S|}$.

Charikar [11] shows that by using degree $N(v)$ of each node v as the *peeling coefficient*, the peeling algorithm, run on a materialized relational graph, can achieve a 2-approximation guarantee for edge-density optimization. Class `EdgeSCANS` in Listing 2 implements the corresponding peeling algorithm in the SCANS system. The API `coefficient(v, K)` performs function `val` on the KMV sketch of node `v` to obtain an estimation of its degree (Line 3). For each subgraph `S` generated during peeling iterations, the API `density(S, K)` estimates its edge-density as the summation of the degree of each node in `S` divided by 2 times the number of nodes in `S` (Lines 6-8).

```
1  class EdgeSCANS: public SCANS{
2      double coefficient(int v, sketches K) {
3          return val(K[v]);
4      }
5      double density(vector<int> S, sketches K){
6          double den;
7          for(int v: S) den += coefficient(v, K);
8          return den / (2 * S.size());
9      }
10 }
```

Listing 2: Edge-densest Subgraph Discovery in SCANS

Theoretical Analysis. We extend Charikar's result and prove that the sketch-based peeling algorithm gives a $2(1+\epsilon)$-approximation of the densest subgraphs discovery problem with high probability (w.h.p.). In the following, we use S_t to denote the subgraph generated at the t-th peeling iteration.

Lemma 1. *For any* $0 \le t < |V|$, *we have* $\frac{\rho_e(S_t)}{1+\delta} \le \widetilde{\rho}_e(S_t) \le \frac{\rho_e(S_t)}{1-\delta}$ *holds with probability at least* $(1-p)$ *given* $\theta = \Theta(\frac{\Lambda}{\delta k} \log \frac{|V|}{p})$.

Proof Sketch. To prove this lemma, we first show that: for any $\delta, p \in (0,1)$ and any node $v \in S_t$, if $\theta = \Theta(\frac{\Lambda}{\delta k} \log \frac{1}{p})$, we have $(1-\delta) \cdot N_{S_t}(v) \le \widetilde{N}_{S_t}(v) \le (1+\delta) \cdot N_{S_t}(v)$ holds with probability at least $1-p$. Then, by using the union bound over all nodes in S_t, we have $\rho_e(S_t) = \frac{\sum_{u \in S_t} N_{S_t}(u)}{2 \cdot |V_{S_t}|} \ge \frac{\sum_{u \in S_t} \widetilde{N}_{S_t}(u)}{2(1+\delta)|V_{S_t}|} = \frac{\widetilde{\rho}_e(S_t)}{1+\delta}$ and similarly $\rho_e(S_t) \le \frac{\sum_{u \in S_t} \widetilde{N}_{S_t}(u)}{2(1-\delta)|V_{S_t}|} = \frac{\widetilde{\rho}_e(S_t)}{1-\delta}$ holds simultaneously with probability $(1-p)$. □

Theorem 3. *Let* S *denote the subgraph returned by Alg. 2, we have* $\rho_e(S) \ge \frac{\rho_e(S^*)}{2(1+\epsilon)}$ *with probability at least* $(1-p)$ *given* $\theta = \Theta(\frac{\Lambda}{\epsilon k} \log \frac{|V|}{p})$.

Proof. For any node $v \in S^*$, since S^* is the densest subgraph of H, we have $\rho_e(S^*) \ge \rho_e(S^* - \{v\})$, i.e. $\frac{|E_{S^*}|}{|V_{S^*}|} \ge \frac{|E_{S^*}| - N_{S^*}(v)}{|V_{S^*}|-1}$. Thus, we have $N_{S^*}(v) \ge \frac{|E_{S^*}|}{|V_{S^*}|} = \rho_e(S^*)$. Consider the iteration when the first node $u \in S^*$ is going to be peeled and let S_t denote the subgraph just before removing u. According to Lemma 1, for any node $v' \in S_t$, we have

$$N_{S_t}(v') \ge \frac{\widetilde{N}_{S_t}(v')}{(1+\delta)} \ge \frac{\widetilde{N}_{S_t}(u)}{(1+\delta)} \ge \frac{(1-\delta)}{(1+\delta)} \cdot N_{S_t}(u) \qquad (2)$$

holds with probability at least $(1-p)$ given $\theta = \Theta(\frac{\Lambda}{\delta k} \log \frac{1}{p})$. The second inequality in (2) holds since the nodes in H are peeled in

the ascending order of estimated degrees. Then, by applying union bounds over all nodes in S_t, we have

$$\rho_e(S_t) = \frac{\sum_{v' \in S_t} N_{S_t}(v')}{2|V_{S_t}|} \geq \frac{(1-\delta)\sum_{v' \in S_t} N_{S_t}(u)}{2(1+\delta)|V_{S_t}|} \geq \frac{(1-\delta)}{2(1+\delta)}\rho_e(S^*) \tag{3}$$

holds with probability at least $(1-p)$ given $\theta = \Theta(\frac{\Lambda}{\delta k}\log\frac{|V|}{p})$.

Let $S = \arg\max_{0 \leq t \leq |V|} \widetilde{\rho}_e(S_t)$ denote the subgraph returned by Alg. 2, according to Lemma 1, we have

$$\rho_e(S) \geq (1-\delta)\widetilde{\rho}_e(S) \geq (1-\delta)\widetilde{\rho}_e(S_t) \geq \frac{1-\delta}{1+\delta}\rho_e(S_t) \geq \frac{(1-\delta)^2}{2(1+\delta)^2}\rho_e(S^*) \tag{4}$$

For any $\epsilon \in (0,1)$, by setting $\delta = (3-2\sqrt{2})\epsilon < 1 + \frac{2-2\sqrt{1+\epsilon}}{\epsilon}$, i.e. $\theta = \Theta(\frac{\Lambda}{\epsilon k}\log\frac{|V|}{p})$, we have $\frac{1}{2(1+\epsilon)} < \frac{(1-\delta)^2}{2(1+\delta)^2}$ and thus $\rho_e(S) > \frac{\rho_e(S^*)}{2(1+\epsilon)}$ with probability at least $(1-p)$. The theorem is proved. □

The time complexity of edge-density optimization with SCANS system is $O(k(\theta+\theta^*)(|\mathcal{E}^*|+|V|k+|V|\log|V|))$ according to Theorem 1 since peeling coefficient estimation requires $f(k) = O(k)$ time to find the largest random number in the KMV sketch. In practice, θ^* is small when optimizing the edge-density metric (see Sec. 5.2). The intuition behind this is the following. The KMV sketch of a node will only be updated during those peeling iterations when the removed node happens to correspond to random numbers in the KMV sketch. For a node v with large degree, the update of its KMV sketch will only take place with small probability $k/N(v)$ due to their large neighborhood. On the other hand, for nodes with small degrees, since nodes are removed in the ascending order of degrees, these nodes will be removed during early peeling iterations and thus also do not require sketch reconstruction w.h.p.

4.2 Triangle-density $\rho_\Delta(S)$

We then further deploy the SCANS system for the optimization of triangle-density [42], which is the number of triangles contained in the subgraph divided by the number of nodes in the subgraph.

Definition 8 (Triangle-density). *For a graph $S = (V_S, E_S)$, its triangle-density is $\rho_\Delta(S) = \frac{\Delta(S)}{|V_S|}$, where $\Delta(S)$ is the number of triangles in S.*

Tsourakakis [42] shows that by using the number of triangles containing the node v as the *peeling coefficient*, denoted as $\Delta(v)$, the peeling algorithm can achieve a 3-approximation guarantee for triangle-density optimization. Thus, to deploy our SCANS system on triangle-density metric, we need to estimate the number of triangles containing each node based on the sketches. The following lemma indicates that the peeling coefficient $\Delta(v)$ can be computed based on the neighborhood of v.

Lemma 2. *For any node $v \in S$, we have $\Delta(v) = \frac{\sum_{u \in N(v)} \Delta(v,u)}{2}$, where $\Delta(v,u)$ denotes the support (u,v), i.e., the number of triangles containing the edge (u,v).*

Based on Lemma 2 and the Horvitz-Thompson [24] estimator, the peeling coefficient $\Delta(v)$ can be estimated through Monte-Carlo simulation. Specifically, if we uniformly sample a set of nodes C from $\mathcal{N}(v)$, the peeling coefficient $\Delta(v)$ can be estimated as $\frac{1}{2}\sum_{u \in C}\frac{\Delta(u,v)}{\frac{|C|}{N(v)}} = \frac{N(v)}{2|C|}\sum_{u \in C}\Delta(u,v)$, where $\frac{|C|}{N(v)}$ is the probability of sampling specific node u from $\mathcal{N}(v)$. The challenge is that since we do not materialize the relational graph, we cannot obtain the exact value of $N(v)$ and $\Delta(u,v)$ for certain sampled neighbor u.

To this end, we provide a novel estimator for $\Delta(v)$ by replacing the $N(v)$ and $\Delta(u,v)$ in the Horvitz-Thompson estimator with corresponding approximate values estimated through the KMV sketches. Specifically, note that the KMV sketch $\mathcal{K}(v)$ of node v offers a uniform sample of its neighbors of size k naturally, i.e., the set of nodes corresponding to the random numbers in $\mathcal{K}(v)$. In the following, we abuse the notation and use $u \in \mathcal{K}(v)$ to denote a node u that is sampled into the KMV sketch. Then, for each node $u \in \mathcal{K}(v)$, since the support of edge (v,u) equals the number of common neighbors of u and v, i.e., $\Delta(u,v) = |\mathcal{N}(u) \cap \mathcal{N}(v)|$, we can estimate $\Delta(u,v)$ for each node $u \in \mathcal{K}(v)$ by taking the intersection of KMV sketches $\mathcal{K}(u)$ and $\mathcal{K}(v)$. Formally, we provide the following estimator for $\Delta(v)$ estimation:

$$\tilde{\Delta}(v) = \frac{\tilde{N}(v) \cdot \sum_{u \in \mathcal{K}(v)} \tilde{\Delta}(u,v)}{2k} \tag{5}$$

where $\tilde{N}(v)$ is the degree estimation of v based on the KMV sketch $\mathcal{K}(v)$. Since the KMV sketch provides a uniform sample from $\mathcal{N}(v)$ of size k, the probability for a certain node u to be sampled is estimated as $\frac{k}{|N(v)|}$. Each $\tilde{\Delta}(u,v)$ is the estimation obtained by taking the intersection of $\mathcal{K}(v)$ and $\mathcal{K}(u)$.

To further obtain the estimator for the triangle density of each subgraph S generated during the peeling iterations, observing that $\rho_\Delta(S) = \frac{\sum_{v \in S}\Delta(v)}{3}$, the estimator for $\rho_\Delta(S)$ can be defined as:

$$\tilde{\rho}_\Delta(v) = \frac{\sum_{v \in S}\tilde{\Delta}(v)}{3} \tag{6}$$

```
1  class TriangleSCANS{
2      virtual double coefficient(int v, sketches K) {
3          double d = val(K[v]), t = 0;
4          for(int u: K[v].nbr()) t += val(K[u] & K[v]);
5          return (t * d) / (K[v].size() * 2);
6      }
7      virtual double density(vector<int> S, sketches K) {
8          double t = 0;
9          for(int v: S) t += coefficient(v, K);
10         return t / (S.size() * 3);
11     }
12 }
```

Listing 3: Triangle-densest Subgraph Discovery in SCANS

Listing 3 gives the implementation of the SCANS system for triangle-densest subgraph discovery. The operator & (Line 4) is reloaded as the intersection between two sketches. The function `coefficient(v, K)` first estimates the degree of node v (Line 3), and then for each neighbor of v sampled within the KMV sketch `K[v]`, it counts the number of triangles containing edge (u,v) based on KMV sketch intersections and records the total number of triangles in variable `t` (Line 4). Finally, the estimator is computed according to Eq. 5 and returned (Line 5). The API `density(S, K)` is implemented to estimate the triangle-density of a given subgraph `S` according to Eq. 6 by calling `coefficient` for each node in `S`.

Theoretical Analysis. We note that the intersection operation required by estimation of $\Delta(v)$ breaks the probabilistic bounds of our estimators. We can nevertheless prove that the estimator in Eq. 5 and 6 provide an unbiased estimation of the *peeling coefficient* of each node accessed and triangle-density of each subgraph generated during the peeling iterations.

Theorem 4. *Given any node $v \in S_t$, we have $\tilde{\Delta}(v)$ is an unbiased estimator of $\Delta(v)$, i.e., $\Delta(v) = \mathbb{E}[\tilde{\Delta}(v)]$.*

Proof. By estimating the degree $\tilde{N}(v)$ and $\tilde{\Delta}(u,v)$ using two different KMV sketches of node v with independent basis, the estimation $\tilde{N}(v)$ and $\sum_{u\in\mathcal{K}(v)} \tilde{\Delta}(u,v)$ are independent to each other. Thus, we have

$$\begin{aligned}\mathbb{E}[\tilde{\Delta}(v)] &= \mathbb{E}[\mathbb{E}[\tilde{\Delta}(v)]] = \mathbb{E}[\frac{\mathbb{E}[\tilde{N}(v)] \cdot \mathbb{E}[\sum_{u\in\mathcal{K}(v)} \tilde{\Delta}(u,v)]}{2k}] \\ &= \mathbb{E}[\frac{N \cdot \sum_{u\in\mathcal{K}(v)} \Delta(u,v)}{2k}] = \Delta(v)\end{aligned} \tag{7}$$

The second equality is due to the independence between $\tilde{N}(v)$ and $\sum_{u\in\mathcal{K}(v)} \tilde{\Delta}(u,v)$; the third equality is due to the unbiasedness of estimation based on the KMV sketch according to [6]; the last equality is due to the Horvitz–Thompson estimator. □

Corollary 1. *For any subgraph S_t generated during the peeling iteration, we have $\tilde{\rho}_\Delta(S_t)$ is an unbiased estimator of $\rho_\Delta(S_t)$, i.e., $\rho_\Delta(S_t) = \mathbb{E}[\tilde{\rho}_\Delta(S_t)]$.*

Proof. This corollary follows trivially from the addition rule of expectation and the unbiasedness of the peeling coefficient estimator in Theorem 4. □

The time complexity of triangle-density optimization with SCANS system is $O(k(\theta+\theta^*)(|\mathcal{E}^*| + |V|k^3 + |V|\log|V|))$ since peeling coefficient estimation has complexity $f(k) = O(k^3)$. Specifically, computing the intersection of two KMV sketches has complexity $O(k^2)$, and k intersections are required since k neighbors are sampled in the KMV sketch. We note that we empirically observed in Sec. 5.5 that θ^* is small when optimizing the triangle-density metric. Although nodes are removed in ascending order wrt. the number of triangles containing them instead of degrees, we can still expect a small θ^* in practice since the triangle number is positively correlated with degree.

4.3 p-mean density $\rho_p(S)$

We then deploy our SCANS system for the optimization of p-mean density, which is first proposed by Veldt et al. [43] as a generalization of edge-density.

Definition 9 (p-mean density). *Given a graph $S = (V_S, E_S)$ and parameter $p \in \mathbb{R}$, the p-mean density of S is defined as $\rho_p(S) = (\frac{1}{|V_S|}\sum_{v\in S} N_S(v)^p)^{1/p}$.*

According to the definition of the peeling coefficient in Eq. 1, we can obtain the corresponding peeling coefficient for p-mean density as:

$$\varphi(u) = N(u)^p + \sum_{v\in\mathcal{N}(u)} (N(v)^p - (N(v)-1)^p) \tag{8}$$

Veldt et al. [43] proved that for any $p \geq 1$, the peeling algorithm based on the above peeling coefficient can yield a $(1+p)^{1/p}$-approximation densest subgraph in terms of p-mean density.

To deploy our SCANS system on p-mean density metric, we provide the following estimator for $\varphi_S(u)$ based on KMV sketches by adapting the Horvitz-Thompson estimator:

$$\tilde{\varphi}(u) = \tilde{N}(u)^p + \frac{\tilde{N}(u) \cdot \sum_{v\in\mathcal{K}(u)} (\tilde{N}(v)^p - (\tilde{N}(v)-1)^p)}{k} \tag{9}$$

The estimator for the p-mean density of each subgraph S generated during the peeling iterations can be defined as:

$$\tilde{\rho}_p(S) = (\frac{1}{|V_S|}\sum_{v\in S} \tilde{N}_S(v)^p)^{1/p} \tag{10}$$

List 4 gives the implementation of the SCANS system for p-mean densest subgraph discovery based on the estimators in Eq. 9 and Eq. 10.

```
class PmeanSCANS: public SCANS{
    double coefficient(int v, sketches K) {
        double d = val(K[v]), t = 0;
        for(int u: K[v].nbr())
            t += (pow(val(u), p) - pow(val(u)-1, p));
        return pow(d, p) + d * t / K[v].size();
    }
    double density(vector<int> S, sketches K){
        double density;
        for(int v: S) density += pow(val(K[v]), p);
        return pow(density / S.size(), 1 / p);
    }
}
```

Listing 4: p-mean densest Subgraph Discovery in SCANS

4.4 f-density $\rho_f(S)$

Finally, we discuss how to deploy SCANS system for the optimization of f-density, which generalize the edge-density in another direction.

Definition 10. *Given a subgraph $S = (V_S, E_S)$ and a monotonically non-decreasing function $f : \mathbb{N}^+ \to \mathbb{R}$ with $f(0) = 0$, the f-density of S is defined as $\rho_f(S) = \frac{|E_S|}{f(|V_S|)}$.*

Kawase and Miyauchi [29] proved that when the function f is either convex or concave, the peeling algorithm with degree as peeling coefficient can be used for f-density optimization with certain approximation guarantees. Class `FSCANS` in Listing 5 implements the corresponding peeling algorithm in the SCANS system. The API `coefficient(v, K)` returns the estimated degree of node `v`. For each subgraph `S`, the API `density(S, K)` estimates its f-density as the summation of the degree of each node in `S` divided by 2 times $f(|V_S|)$.

5 EXPERIMENTS

5.1 Experimental Setup

Environment. All experiments are conducted on a Linux Server with AMD EPYC 7643 CPU and 256 GB memory running Ubuntu

```
class FSCANS: public SCANS{
    double coefficient(int v, sketches K) {
        return val(K[v]);
    }
    double density(vector<int> S, sketches K){
        double den;
        for(int v: S) den += coefficient(v, K);
        return den / 2 * f(S.size());
    }
}
```

Listing 5: f-densest Subgraph Discovery in SCANS

Table 1: Statistics of KGs used in the experiments, where $|\mathcal{L}(\mathcal{V})|$ and $|\mathcal{L}(\mathcal{E})|$ denote the numbers of node and edge types, and $|\mathbb{M}|$ is the number of evaluated meta-paths.

Dataset	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{L}(\mathcal{V})\rvert$	$\lvert\mathcal{L}(\mathcal{E})\rvert$	$\lvert\mathbb{M}\rvert$
IMDB	21.4K	86.6K	4	6	679
ACM	10.9K	548K	4	8	20
DBLP	26.1K	239.6K	4	6	48
PubMed	63.1K	236.5K	4	10	172
FreeBase	180K	1.06M	8	36	151
TPCH	1.85M~29.6M	7.5M~120M	3	2	1

20.04. All algorithms are implemented in C++ with `-O3` optimization and executed in a single thread.

Dataset. We conduct our experiments on six real-world datasets. The statistics of those KGs are reported in Table 1. IMDB is a KG about actors and directors of movies. ACM and DBLP are two academic KGs denoting the co-authorships between researchers through papers and publishing venues. PubMed is a biomedical KG representing the relationships between genes, chemicals, and diseases of different species. FreeBase is extracted from a general-purpose KG developed by Google[2]. Besides, we also conduct the scalability tests on synthetic TPCH datasets [1] with various scale factors.

Query Generation. We use AnyBURL [32] to extract a set of candidate meta-paths $\mathbb{M}$. We set the confidence threshold to 0.1 for AnyBURL and take the snapshot at ten seconds. Column $|\mathbb{M}|$ in Table 1 shows the number of meta-paths found on each dataset.

Compared Methods. We compare peeling algorithms implemented through our SCANS system to algorithms based on explicit relational graph materialization.

- FlowE [21] and FlowT [34] apply the max-flow method to find the optimal densest subgraph given the materialized relational graph in terms of edge-density and Δ-density respectively.
- PeelE [11] and PeelT [42] apply the peeling paradigm to find the approximate densest subgraph given the materialized relational graph in terms of edge-density and Δ-density respectively.
- ScansE and ScansT are peeling algorithms implemented through SCANS in terms of edge-density and Δ-density respectively.

Parameter Setting. In our experiments, we set $k = 24$, $k^- = 4$ and $\theta = 1$ for both ScansE and ScansT by default. We choose the default values of hyperparameters to ensure that our methods ScansE and ScansT returns a subgraph with corresponding density larger than 0.95 of densities achieved by methods based on materialized relational graphs on average across datasets (Sec. 5.3). These settings demonstrate that peeling algorithms based on SCANS system can yield accurate results in practice with significantly less stringency than those outlined in the worst-case analysis.

Evaluation Metrics. 1) For efficiency, we report the average running time of each algorithm to process the densest subgraph discovery problem per meta-path. 2) For effectiveness, since the exact methods cannot terminate in a reasonable time for most datasets, we quantify the quality of our methods by the ratio γ between the density of the subgraph returned by our methods (ScansE and ScansT) and the density achieved by the corresponding materialization-based peeling algorithms (PeelE and PeelT).

5.2 Efficiency Evaluation

Time Consumption. In Table 2, we present the execution time per query for each method. We have the following observations.

First, both methods ScansE and ScansT based on SCANS system is much more efficient than materialization-based methods. Materialization-based methods cannot handle all experimental cases in reasonable time. Flow-based methods can only handle cases on small datasets (FlowE cannot terminate in 24 hours except for datasets IMDB and ACM, FlowT can only handle queries from IMDB) due to the time-prohibitive flow computations. Materialization-based peeling algorithm PeelT cannot handle queries optimizing triangle-density from PubMed, DBLP and FreeBase due to the time consuming triangle enumeration for peeling coefficient computation. On the other hand, our methods can handle all experimental cases efficiently. ScansE and ScansT handle the largest dataset FreeBase within 0.34 and 3.64 seconds per query respectively on average. Compared to materialization-based peeling methods, ScansE achieves upto 53.4 times speedup than PeelE on PubMed, and ScansT achieves upto 1063 times speedup than PeelT on ACM.

Second, methods based on SCANS system are more robust to complex density metrics than materialization-based peeling algorithms in terms of efficiency. For method PeelE optimizing edge-density metric, we can observe that the relational graph materialization stage dominates its time consumption. The materialization time consistently occupies more than 85% of total time of PeelE across datasets and upto 99.58% on dataset ACM. However, this ratio drops significantly for PeelT optimizing triangle-density metrics, which are much more time-consuming than PeelE. Specifically, on dataset ACM, PeelE handle queries optimizing edge-density metric within 3.0141 seconds, whereas the corresponding method PeelT targets queries optimizing triangle-density metric requires 743.11 seconds per query on average due to the time consuming triangle enumeration. On the other hand, our method ScansT can still handle queries optimizing triangle-density metric efficiently. The reason is that based on the KMV sketches, our methods ScansT restricts the time consuming computations of peeling coefficient on the KMV sketches of limited sizes and thus the running time becomes less sensitive to the density metrics.

Overall, SCANS can accelerate the densest subgraph discovery by simultaneously avoiding the dominating relational graph materialization stage and also circumventing the time-consuming peeling coefficient computation.

Memory Usage. Fig. 3 compares the memory usage of the materialized relational graph required by PeelE and PeelT with the KMV sketches utilized by ScansE and ScansT. For each dataset, we report

[2]https://developers.google.com/freebase

Table 2: Results for the efficiency of different algorithms. For exact algorithms FlowE and FlowT, the returning time of each dataset is reported. For PeelE and PeelT, we report the time consumption as well as the proportion of relational graph materialization. For ScansE and ScansT, we additionally report the speedup ratios over PeelE and PeelT. The number of reconstructed sketches in peeling iterations for optimizing ρ_e and ρ_Δ are reported as θ_e^* and θ_Δ^* respectively. Cells with a dash line denote that the method cannot be finished within 24 hours.

Metric	ρ_e						ρ_Δ					
Method	FlowE	PeelE		ScansE		θ_e^*	FlowT	PeelT		ScansT		θ_Δ^*
IMDB	0.0017s	0.0013s	85.06%	0.0008s	1.53x	1.24	0.143s	0.002s	58.58%	0.0018s	1.1x	1.24
ACM	31.603s	3.0141s	99.58%	0.24s	12.6x	4.75	-	723.11s	0.435%	0.68s	1063x	4.65
DBLP	-	4.8328s	97.67%	0.1554s	31x	10.02	-	-	-	1.55s	-	10.48
PubMed	-	4.5380s	97.34%	0.085s	53.4x	5.22	-	-	-	1.01s	-	5.20
FreeBase	-	18.0464s	96.62%	0.34s	53.1x	8.27	-	-	-	-	-	9.32

the peak memory usage among all meta-path queries. We have the following observations.

First, the additional memory usage required by the KMV sketches are consistently smaller than the materialized relational graph across datasets. Besides, the gap between memory usage of KMV sketches and materialized relational graphs increases with the increase of sizes of datasets. Specifically, on small datasets IMDB, KMV sketches requires 7.7 times less memory the the materialized relational graphs, whereas on the large dataset FreeBase, KMV sketches requires upto 471 times less additional memory than materializing the relational graphs explicitly. The reason is that the memory usage of KMV sketches are linear to the number of nodes in the dataset, whereas the materialized relational graph consumes upto the square of number of nodes in the dataset in the worst case.

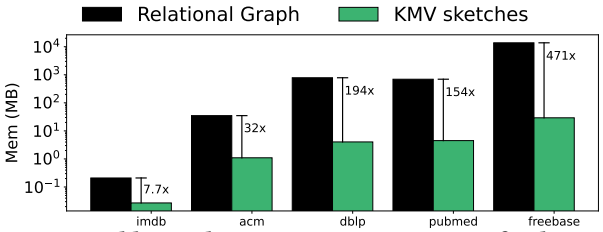
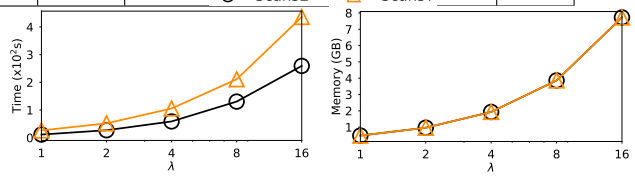


Figure 3: Additional memory consumption of relational graphs materialized in PeelE/PeelT, and KMV sketches constructed in ScansE/ScansT.

Scalability. In this section, we conduct a scalability test of methods based on SCANS system through experiments over large synthetic datasets generated by the TPCH benchmark with scale factors $\lambda \in \{1, 2, 4, 8, 16\}$. We conduct our experiments with the meta-path (customer)→ (order)→(product), which includes a relational graph connecting customers who have bought the same product [46]. We can observe that both the running time and the memory usage of KMV sketches are linear to the size of the synthetic dataset, which verified the scalability of SCANS based methods. Specifically, our method ScansE and ScansT can handle the largest TPCH dataset with around 30M nodes within 260 seconds and 436 seconds respectively with 7.72GB additional memory for KMV sketches.

Parameter Analysis. We further study the influence of the KMV sketch parameters over the efficiency of our algorithm. Fig. 5(a)-Fig. 5(e) reports the running time of method ScansE varying parameters k and θ across datasets. We can observe that with the increase of parameter k, the time consumption of ScansE decreases monotonically across datasets except ACM, where the time consumption of ScansE first decrease when k is varied from 8 to 12 and then



(a) Running Time (b) Memory Usage

Figure 4: The running time and memory usage of the KMV sketches for methods ScansE and ScansT.

increase with the further increase of k. The reason is that with the increase of k, the number of KMV sketches reconstruction θ^* decreases and thus makes the algorithm less time-consuming (as displayed in Fig. 5(f)–5(j)). However, when the value k becomes too large (larger than 12 on ACM), the time for KMV sketches construction and peeling coefficient estimation based on KMV sketches become dominating and thus the time consumption again increases with the value of k. On the other hand, with the increase of θ, the time consumption of ScansE increases significantly. The reason is that the larger value of θ increases the time for KMV sketches construction and peeling coefficient estimation and increases the number of sketches reconstructed θ^* during the peeling iterations.

For method ScansT, the value k influences its time efficiency in an opposite direction. Fig. 5(k)– 5(o) reports the running time of method ScansE varying parameters k and θ across datasets. We can observe that with the increase of sketch size k, the running time of ScansT increases monotonically across datasets except FreeBase, even though the number of reconstructed sketches θ^* follows the same distribution in ScansE (as displayed in Fig. 5(p)–5(t)). The reason is that the time consumption of peeling coefficient computation for triangle-density metric is influenced by k more significantly. As discussed in Sec. 4.2, the time complexity of method ScansT is $O(k^4)$ due to the estimation of number triangles containing each node. Thus, the time consumption for peeling coefficient estimation dominates the running time of ScansT, which increases with the increase of k.

Fig. 6 reports the running time of and number of sketches reconstructed in ScansE and ScansT varying parameter k^- across datasets. We can observe that both the time consumption of ScansE and ScansT grows exponentially with the increase of parameter k^- due to the exponential increase of number of sketches reconstructed during the peeling iterations. Thus, we set $k^- = 4$ by default to achieve good efficiency of our methods during the experimentation.

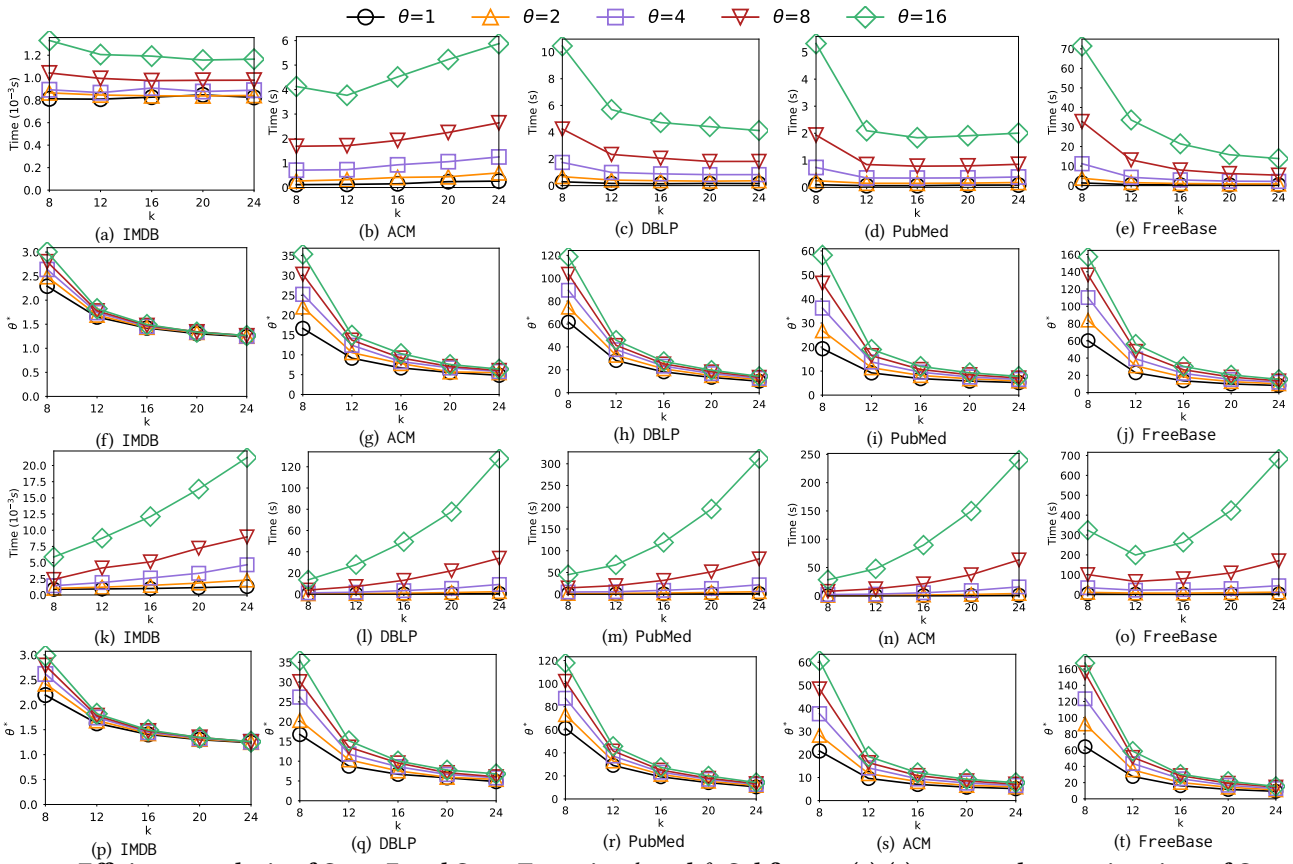


Figure 5: Efficiency analysis of ScansE and ScansT varying k and θ. Subfigures (a)-(e) reports the running time of ScansE; subfigures (f)-(j) reports the number of reconstructed sketches during peeling iterations of ScansE; subfigures (k)-(o) reports the running time of ScansT; subfigures (p)-(t) reports the number of reconstructed sketches during peeling iterations of ScansT.

5.3 Effective Evaluation

Fig. 7 reports the average ratio γ between the edge-density of subgraph returned by ScansE and the edge-density of subgraphs returned by PeelE across datasets varying parameters k and θ.

We first observe that the value γ increases monotonically with the increase of both k and θ across all datasets and approaches to 1, which verifies Theorem 3 that ScansE provides a $(2+\epsilon)$-approximation to the DSD problem based on edge-density ρ_e.

Furthermore, the effectiveness of ScansE is affected by the sketch size k more significantly than by the number of sketches θ. Thus, in order to achieve better effectiveness, the users may prefer to increase the sketch size k instead of increasing the number of sketches θ. For example, in FreeBase, increasing k from 8 to 16 fixing $\theta = 1$ boost γ from 90% to 95%, whereas increasing θ from 1 to 2 fixing $k = 8$ boosts γ from 90% to 93%. The reason is that increasing the value of θ will increase the number of KMV sketch reconstructions during the peeling iterations, which can make the estimation less accurate due to the independence of the reconstructed KMV sketches to the previous KMV sketches. Following this observation, we choose the default value of $k = 24$ and $\theta = 1$, which ensures that ScansE achieves $\gamma > 95\%$ across all datasets. For method ScansT optimizing the triangle-density, since PeelT cannot finish in reasonable time on datasets except IMDB and ACM, we only compare the effectiveness of ScansT with PeelT on IMDB and ACM. On IMDB, ScansT returns a subgraph with 99.65% triangle-density to the subgraph returned by PeelT, and 95.7% on ACM.

Table 3: The ratio between the size of densest subgraph discovered by ScansE and PeelE.

Dataset	IMDB	ACM	DBLP	PubMed	FreeBase
Size ratio	101.6%	102.9%	102.1%	101.9%	102.5%

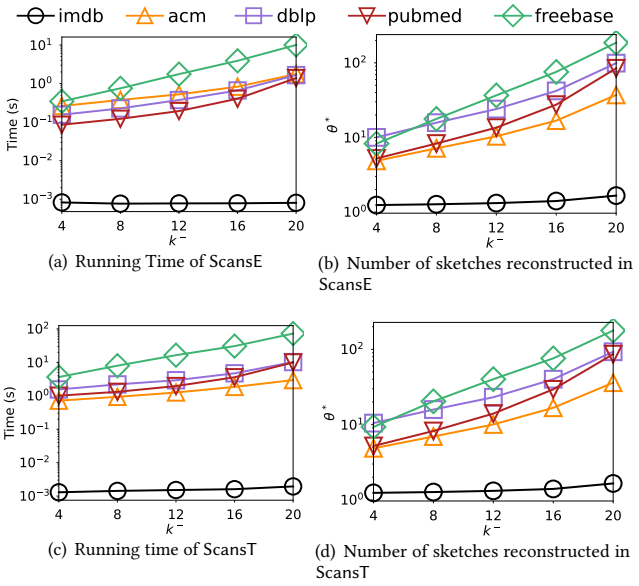
Table 3 reports the average ratio between the size of densest subgraphs discovered by our method ScansE and the materialization-based peeling method PeelE. We can observe that our method ScansE reveals densest subgraphs with comparable sizes (consistently less than 3% relative differences across datasets) to the densest subgraphs discovered by PeelE.

Overall, by setting the default value of $k = 24$ and $\theta = 1$, both our methods ScansE and ScansT can achieve 95%-approximation and comparable sizes to the subgraphs returned by the corresponding materialization-based peeling methods.

5.4 Case Study

To further verify the effectiveness of SCANS over real-world applications, we apply the ScansE and ScansT methods on heterogeneous data generated from our industrial collaborator Grab, which is one of the leading technology companies known for its diverse services including digital payments and food delivery.[3] We obtain three

[3]https://www.grab.com/sg/



(a) Running Time of ScansE

(b) Number of sketches reconstructed in ScansE

(c) Running time of ScansT

(d) Number of sketches reconstructed in ScansT

Figure 6: Efficiency Analysis of ScansE and ScansT varying parameter k^- across datasets. Subfigure (a) and (b) reports the running time of and number of sketches reconstructed in ScansE; subfigure (c) and (d) reports the corresponding evaluation metrics of ScansT.

different types of nodes *account*(A), *merchant*(M), *device*(D) and two types of edges account→merchant ("order"), account→device ("login"). We choose the meta-path "(account) → (device)" induces a relational graph connecting two accounts if they login through the same device. Such relational graph tends to form dense subgraphs containing fraudulent accounts. For each account in the dataset, it contains a ground-truth label denoting whether this account is a fraudster. In scenarios such as artificially inflating a merchant's ranking and visibility through bulk ordering, fraudsters typically create numerous accounts to post positive reviews. These fraudulent actions often lead to the creation of a significant number of accounts, each logging in once, which results in a sparse representation in the heterogeneous graph, making many fraudulent accounts undetectable by DSD. However, if we detect on the relational graph, as long as these accounts are connected through shared devices, they will form dense subgraphs on the relational graph.

Figure 8 illustrates the fraudulent community identified by our method, ScansE. This community was not detectable by traditional DSD methods due to its sparse nature in the original heterogeneous graph. The subgraph uncovered by ScansE on the induced relational graphs is composed of two connected components. Of the 195 accounts flagged by ScansE, 190 are genuine fraudulent accounts, with 5 being normal accounts (false positives), resulting in a precision of 97.43%. Similarly, the triangle-density-based method, ScansT, identifies the same two connected components as ScansE, along with an additional component comprising 9 fraudulent accounts, thus achieving a slightly higher precision of 97.55%.

5.5 Synthetic Analysis of θ^*

On top of reporting the parameter θ^* for the real-world datasets, we also perform an analysis of θ^* over synthetic graphs generated with different models to explore possible factors that influence the number of sketch reconstructions in peeling algorithms based on the SCANS system. Specifically, we examine the random binomial graphs and the scale-free graphs generated with Erdős–Rényi (ER) model and Barabási–Albert (BA) model respectively. Fig. 9 reports the value of θ^* varying the number of nodes $|V|$ in the graph exponentially from 1000 to 16000 and we make the following observation. The value of θ^* grows logarithm with the number of nodes $|V|$ in the synthetic graphs on both binomial graphs (Fig. 9(a) and 9(b)) and scale-free graphs (Fig. 9(c) and 9(d)) for both methods ScansE and ScansT optimizing the edge-density and triangle-density respectively. Based on this observation, we proposed the following hypothesis that the number of reconstructed sketches required by the peeling algorithm ScansE and ScansT are logarithm to the number of nodes in the graph regardless of the specfic structure of the relational graphs.

Hypothesis 1. *Given a relational graph $H = (V, E)$, we have $\theta^* = O(\log |V|)$ for methods ScansE and ScansT.*

We leave the proof of this hypothesis as a future work.

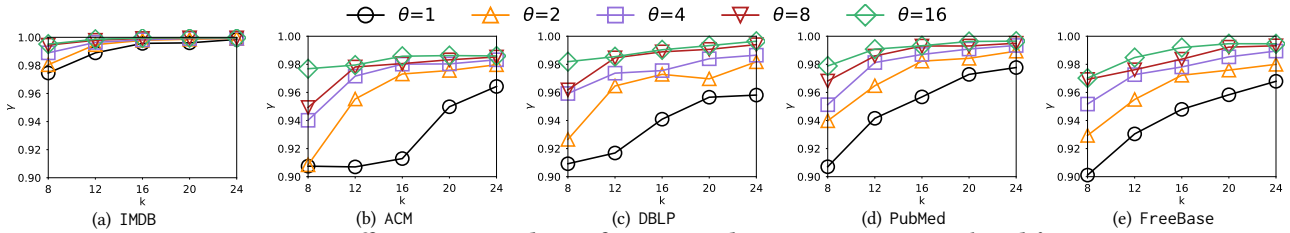
6 RELATED WORK

Densest Subgraph Discovery. Previous studies have extensively explored the problem of densest subgraph discovery. Exact algorithms for densest subgraph discovery involve solving a max-flow [17, 21] or linear programming [11] problem, which is not scalable to very large graphs. Thus, the peeling paradigm, which is efficient with approximation guarantees, has been widely adopted in the literature. Charikar [11] proposed the peeling algorithm for edge-density, which iteratively removes nodes with smallest degree and returns the subgraph with largest edge-density generated during the peeling iterations. Instead of removing one node in each peeling iteration, Bahmani et al. [3] proposed to remove all nodes with degree less than $2(1+\epsilon)\rho$ at each peeling iteration and prove that this algorithm achieves a solution of $2(1+\epsilon)$-approximation. Boob et al. [8] proposed the multi-round peeling algorithm, which obtains the densest subgraph by iteratively removing the node with the smallest load, where the load of a node in each round is the sum of its induced degree and its load in previous round. The peeling paradigm has also been extended to other density metrics. Tsourakakis et al. [42] modeled the graph density as the number of k-cliques contained in the graph and showed that the peeling algorithm of repeatedly removing the node contained in the smallest number of corresponding cliques achieves k-approximation.

All these works assume that the relational graph is already materialized and thus cannot scale to large heterogeneous data sources owing to the requirement of materializing the relational graph.

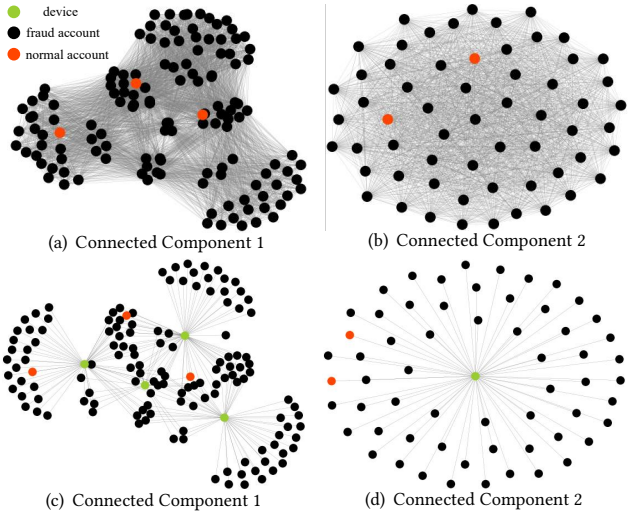
Community Search over Heterogeneous Information Networks (HIN). Recently, several works have focused on the community search problem over HINs based on meta-paths.

Fang et al. [16] proposed to reveal $(k, \mathcal{M})$-core from HINs, which is equivalent to the k-core in the relational graph induced by meta-path $\mathcal{M}$. The algorithm dynamically maintains k neighbors for each node in the relational graph through meta-path instances and removes nodes with less than k neighbors iteratively. Yang et al. [49] introduced the $(k, \mathcal{M})$-Btruss and $(k, \mathcal{M})$-Ctruss models for community search in KGs. The $(k, \mathcal{M})$-Btruss is equivalent to



(a) IMDB (b) ACM (c) DBLP (d) PubMed (e) FreeBase

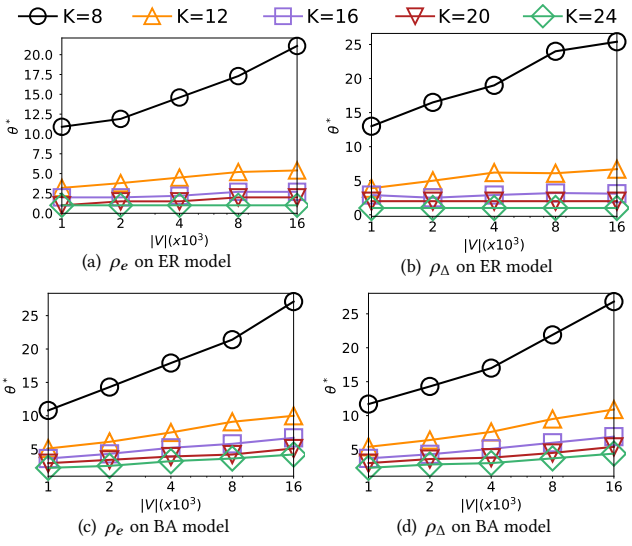
Figure 7: Effectiveness analysis of ScansE with varying parameters k and θ.



(a) Connected Component 1 (b) Connected Component 2

(c) Connected Component 1 (d) Connected Component 2

Figure 8: Densest subgraph discovered by ScansE on the relational graph induced by the meta-path "(account) → (device)". (a) and (b) denote the two connected components of the densest subgraph; (c) and (d) denote the corresponding subgraph in the heterogeneous source.



(a) ρ_e on ER model (b) ρ_Δ on ER model

(c) ρ_e on BA model (d) ρ_Δ on BA model

Figure 9: Number of sketches reconstructed θ^* during peeling iteration with varying sizes of synthetic graphs.

the k-truss in the relational graph induced by $\mathcal{M}$, whereas $(k, \mathcal{M})$-Ctruss considers the overlaps between meta-path instances and does not correspond to existing models of cohesive subgraphs in the relational graphs. Jiang et al. [26] studied the nested meta-path core (k, Ψ)-core, which aims at finding a subgraph that is $(k, \mathcal{M})$-core for each meta-path $\mathcal{M} \in \Psi$.

The above methods focus on discovering subgraphs conforming to certain topological community models with strict topological constraints and can overlook relevant subgraphs with complex topological structures [50]. Instead, our work focuses on finding the subgraph optimizing the density metric, which is more flexible than topology-driven community search methods.

Strouthopoulos and Papadopoulos [38] studied the discovery of k-cores in hidden networks, where the existence of any edge can only be decided via a *probe operation*. However, this work also aims at finding subgraphs conforming to the k-core model. Besides, their method cannot handle large relational graphs since the probe operations are time-consuming over relational graphs, as it involves bread-first search (BFS) on the matching graph.

Counting Sketches and Graph Sketches. Sketches have been widely used for streaming algorithms. Various sketches have been proposed for counting distinct values. PCSA [19] uses a bit vector of of length L that is logarithmic in the stream cardinality and hashes each of the distinct values to a binary string in $\{0, 1\}^L$. By keeping track of the position r of the leftmost 0 bit over all of the hashed values, the cardinality is roughly estimated as 2^r. A series of works have been proposed to further optimize the counting sketches including LogLog [15], Hyperloglog [18, 22] and Hyperlogloglog [28]. Other sketches such as count-min sketch and its variants [12, 14, 41, 47, 48] are proposed for frequency estimation over streaming data. Besides count sketches, graph sketches have been proposed for various streaming graph queries such as subgraph counting [5, 9, 27] and pagerank [35]. Nevertheless, these sketch based methods have not been applied for densest subgraph discovery, particularly over unmaterialized relational graphs.

7 CONCLUSION

This paper introduces SCANS, a materialization-free processing system for densest subgraph discovery over relational graphs. Our system, grounded in the *sketch-peeling* approach, facilitates peeling coefficient and subgraph density estimation for peeling iterations directly from heterogeneous data sources. Additionally, we craft user-friendly APIs within the SCANS system, making it simpler for users to implement a variety of peeling algorithms for different density metrics. We establish that SCANS can achieve constant approximation guarantees for densest subgraph discovery based on both edge- and triangle-density. Extensive experiments validate the superiority of the SCANS system over baselines across various datasets based on both edge- and triangle-density metrics.

REFERENCES

[1] [n.d.]. https://www.tpc.org/tpch/.

[2] Daniel J. Abadi, Adam Marcus, Samuel Madden, and Kate Hollenbach. 2009. SW-Store: a vertically partitioned DBMS for Semantic Web data management. *VLDB J.* 18, 2 (2009), 385–406.

[3] Bahman Bahmani, Ravi Kumar, and Sergei Vassilvitskii. 2012. Densest Subgraph in Streaming and MapReduce. *Proc. VLDB Endow.* 5, 5 (2012), 454–465.

[4] Ziv Bar-Yossef, TS Jayram, Ravi Kumar, D Sivakumar, and Luca Trevisan. 2002. Counting distinct elements in a data stream. In *Randomization and Approximation Techniques in Computer Science: 6th International Workshop, RANDOM 2002 Cambridge, MA, USA, September 13–15, 2002 Proceedings 5*. Springer, 1–10.

[5] Ziv Bar-Yossef, Ravi Kumar, and D Sivakumar. 2002. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *SODA*, Vol. 2. 623–632.

[6] Kevin S. Beyer, Peter J. Haas, Berthold Reinwald, Yannis Sismanis, and Rainer Gemulla. 2007. On Synopses for Distinct-Value Estimation under Multiset Operations. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (SIGMOD '07)*. Association for Computing Machinery, New York, NY, USA, 199–210.

[7] Francesco Bonchi, Arijit Khan, and Lorenzo Severini. 2019. Distance-generalized Core Decomposition. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*. ACM, 1006–1023.

[8] Digvijay Boob, Yu Gao, Richard Peng, Saurabh Sawlani, Charalampos E. Tsourakakis, Di Wang, and Junxing Wang. 2020. Flowless: Extracting Densest Subgraphs Without Flow Computations. In *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*. ACM / IW3C2, 573–583.

[9] Luciana S Buriol, Gereon Frahling, Stefano Leonardi, Alberto Marchetti-Spaccamela, and Christian Sohler. 2006. Counting triangles in data streams. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 253–262.

[10] Bokai Cao, Mia Mao, Siim Viidu, and Philip S. Yu. 2017. HitFraud: A Broad Learning Approach for Collective Fraud Detection in Heterogeneous Information Networks. In *2017 IEEE International Conference on Data Mining, ICDM 2017, New Orleans, LA, USA, November 18-21, 2017*. IEEE Computer Society, 769–774.

[11] Moses Charikar. 2000. Greedy approximation algorithms for finding dense components in a graph. In *Approximation Algorithms for Combinatorial Optimization, Third International Workshop, APPROX 2000, Saarbrücken, Germany, September 5-8, 2000, Proceedings (Lecture Notes in Computer Science)*, Vol. 1913. Springer, 84–95.

[12] Moses Charikar, Kevin Chen, and Martin Farach-Colton. 2002. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming*. Springer, 693–703.

[13] Edith Cohen, Eran Halperin, Haim Kaplan, and Uri Zwick. 2002. Reachability and distance queries via 2-hop labels. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA*. ACM/SIAM, 937–946.

[14] Graham Cormode and Shan Muthukrishnan. 2005. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms* 55, 1 (2005), 58–75.

[15] Marianne Durand and Philippe Flajolet. 2003. Loglog counting of large cardinalities. In *Algorithms-ESA 2003: 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003. Proceedings 11*. Springer, 605–617.

[16] Yixiang Fang, Yixing Yang, Wenjie Zhang, Xuemin Lin, and Xin Cao. 2020. Effective and Efficient Community Search over Large Heterogeneous Information Networks. *Proc. VLDB Endow.* 13, 6 (2020), 854–867.

[17] Yixiang Fang, Kaiqiang Yu, Reynold Cheng, Laks VS Lakshmanan, and Xuemin Lin. 2019. Efficient algorithms for densest subgraph discovery. *Proceedings of the VLDB Endowment* 12, 11 (2019), 1719–1732.

[18] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. 2007. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. *Discrete mathematics & theoretical computer science* Proceedings (2007).

[19] Philippe Flajolet and G Nigel Martin. 1985. Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences* 31, 2 (1985), 182–209.

[20] Aristides Gionis, Flavio Junqueira, Vincent Leroy, Marco Serafini, and Ingmar Weber. 2013. Piggybacking on Social Networks. *Proc. VLDB Endow.* 6, 6 (2013), 409–420.

[21] A. V. Goldberg. 1984. *Finding a Maximum Density Subgraph.* Technical Report. USA.

[22] Stefan Heule, Marc Nunkesser, and Alexander Hall. 2013. Hyperloglog in practice: Algorithmic engineering of a state of the art cardinality estimation algorithm. In *Proceedings of the 16th International Conference on Extending Database Technology*. 683–692.

[23] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan F. Sequeda, Steffen Staab, and Antoine Zimmermann. 2022. Knowledge Graphs. *ACM Comput. Surv.* 54, 4 (2022), 71:1–71:37.

[24] Daniel G Horvitz and Donovan J Thompson. 1952. A generalization of sampling without replacement from a finite universe. *Journal of the American statistical Association* 47, 260 (1952), 663–685.

[25] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. 2022. A Survey on Knowledge Graphs: Representation, Acquisition, and Applications. *IEEE Trans. Neural Networks Learn. Syst.* 33, 2 (2022), 494–514.

[26] Yangqin Jiang, Yixiang Fang, Chenhao Ma, Xin Cao, and Chunshan Li. 2022. Effective Community Search over Large Star-Schema Heterogeneous Information Networks. *Proc. VLDB Endow.* 15, 11 (2022), 2307–2320.

[27] Hossein Jowhari and Mohammad Ghodsi. 2005. New streaming algorithms for counting triangles in graphs. In *Computing and Combinatorics: 11th Annual International Conference, COCOON 2005 Kunming, China, August 16–19, 2005 Proceedings 11*. Springer, 710–716.

[28] Matti Karppa and Rasmus Pagh. 2022. HyperLogLogLog: Cardinality Estimation With One Log More. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 753–761.

[29] Yasushi Kawase and Atsushi Miyauchi. 2018. The Densest Subgraph Problem with a Convex/Concave Size Function. *Algorithmica* 80, 12 (2018), 3461–3480.

[30] Jonathan Kuck, Honglei Zhuang, Xifeng Yan, Hasan Cam, and Jiawei Han. 2015. Query-based outlier detection in heterogeneous information networks. In *Advances in database technology: proceedings. International Conference on Extending Database Technology*, Vol. 2015. NIH Public Access, 325.

[31] Laks V. S. Lakshmanan. 2022. On a Quest for Combating Filter Bubbles and Misinformation. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*. ACM, 2.

[32] Christian Meilicke, Melisachew Wudage Chekol, Daniel Ruffinelli, and Heiner Stuckenschmidt. 2019. Anytime Bottom-Up Rule Learning for Knowledge Graph Completion. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 3137–3143.

[33] Changping Meng, Reynold Cheng, Silviu Maniu, Pierre Senellart, and Wangda Zhang. 2015. Discovering meta-paths in large heterogeneous information networks. In *Proceedings of the 24th international conference on world wide web*. 754–764.

[34] Michael Mitzenmacher, Jakub Pachocki, Richard Peng, Charalampos Tsourakakis, and Shen Chen Xu. 2015. Scalable large near-clique detection in large-scale networks via sampling. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 815–824.

[35] Atish Das Sarma, Sreenivas Gollapudi, and Rina Panigrahy. 2011. Estimating pagerank on graph streams. *Journal of the ACM (JACM)* 58, 3 (2011), 1–19.

[36] Chuan Shi, Zhiqiang Zhang, Ping Luo, Philip S Yu, Yading Yue, and Bin Wu. 2015. Semantic path based personalized recommendation on weighted heterogeneous information networks. In *Proceedings of the 24th ACM international on conference on information and knowledge management*. 453–462.

[37] Sriganesh Srihari and Hon Wai Leong. 2013. A Survey of Computational Methods for protein Complex Prediction from protein Interaction Networks. *J. Bioinform. Comput. Biol.* 11, 2 (2013).

[38] Panagiotis Strouthopoulos and Apostolos N Papadopoulos. 2019. Core discovery in hidden networks. *Data & Knowledge Engineering* 120 (2019), 45–59.

[39] Wen Sun, Achille Fokoue, Kavitha Srinivas, Anastasios Kementsietsidis, Gang Hu, and Guo Tong Xie. 2015. SQLGraph: An Efficient Relational-Based Property Graph Store. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, Timos K. Sellis, Susan B. Davidson, and Zachary G. Ives (Eds.). ACM, 1887–1901.

[40] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. *Proc. VLDB Endow.* 4, 11 (2011), 992–1003.

[41] Lu Tang, Qun Huang, and Patrick PC Lee. 2019. Mv-sketch: A fast and compact invertible sketch for heavy flow detection in network data streams. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2026–2034.

[42] Charalampos E. Tsourakakis. 2015. The K-clique Densest Subgraph Problem. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, Aldo Gangemi, Stefano Leonardi, and Alessandro Panconesi (Eds.). ACM, 1122–1132.

[43] Nate Veldt, Austin R. Benson, and Jon M. Kleinberg. 2021. The Generalized Mean Densest Subgraph Problem. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*. ACM, 1604–1614.

[44] Bryan Wilder, Nicole Immorlica, Eric Rice, and Milind Tambe. 2018. Maximizing Influence in an Unknown Social Network. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. AAAI Press, 4743–4750.

[45] Min Wu, Xiaoli Li, Chee Keong Kwoh, and See-Kiong Ng. 2009. A core-attachment based method to detect protein complexes in PPI networks. *BMC Bioinform.* 10 (2009).

[46] Konstantinos Xirogiannopoulos and Amol Deshpande. 2017. Extracting and Analyzing Hidden Graphs from Relational Databases. In *Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD '17)*. Association for Computing Machinery, New York, NY, USA, 897–912.

[47] Tong Yang, Jie Jiang, Peng Liu, Qun Huang, Junzhi Gong, Yang Zhou, Rui Miao, Xiaoming Li, and Steve Uhlig. 2018. Elastic sketch: Adaptive and fast network-wide measurements. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. 561–575.

[48] Tong Yang, Haowei Zhang, Jinyang Li, Junzhi Gong, Steve Uhlig, Shigang Chen, and Xiaoming Li. 2019. HeavyKeeper: an accurate algorithm for finding Top-k elephant flows. *IEEE/ACM Transactions on Networking* 27, 5 (2019), 1845–1858.

[49] Yixing Yang, Yixiang Fang, Xuemin Lin, and Wenjie Zhang. 2020. Effective and Efficient Truss Computation over Large Heterogeneous Information Networks. In *36th IEEE International Conference on Data Engineering (ICDE)*. IEEE, 901–912.

[50] Niu Yudong, Yuchen Li, Ju Fan, and Zhifeng Bao. 2022. Local Clustering over Labeled Graphs: An Index-Free Approach. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2805–2817.

[51] Yingli Zhou, Yixiang Fang, Wensheng Luo, and Yunming Ye. 2023. Influential Community Search over Large Heterogeneous Information Networks. *Proceedings of the VLDB Endowment* 16, 8 (2023), 2047–2060.