Anchored Densest Subgraph

Yizhou Dai
The University of Auckland
Auckland, New Zealand
ydai992@aucklanduni.ac.nz

Miao Qiao
The University of Auckland
Auckland, New Zealand
miao.qiao@auckland.ac.nz

Lijun Chang
The University of Sydney
Sydney, Australia
Lijun.Chang@sydney.edu.au

ABSTRACT

Given a graph, densest subgraph search reports a single subgraph that maximizes the density (*i.e.*, average degree). To diversify the search results without imposing rigid constraints, this paper studies the problem of *anchored densest subgraph search (*ADS*)*. Given a graph, a reference node set R and an anchored node set A with $A \subseteq R$, ADS reports a supergraph of A that maximizes the *R-subgraph density* – a density that favors nodes that are close to R and are not over-popular compared to nodes in R. These two levels of localities to R bring wide applications, as demonstrated by our use cases. For ADS, we propose an algorithm that is local since the complexity is only related to the nodes in R as opposed to the entire graph. Extensive experiments show that our local algorithm for ADS outperforms the global algorithm by up to three orders of magnitude in time and space consumption; moreover, our local algorithm outperforms existing local community detection solutions in locality, result density, and query processing time and space.

CCS CONCEPTS

• **Mathematics of computing** → **Graph algorithms**; • **Information systems** → *Clustering*.

KEYWORDS

dense subgraph search, local community detection, network flow

ACM Reference Format:
Yizhou Dai, Miao Qiao, and Lijun Chang. 2022. Anchored Densest Subgraph. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22), June 12–17, 2022, Philadelphia, PA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3514221.3517890

1 INTRODUCTION

The interconnections among real-world objects can be modeled as a graph $G(V, E)$ where the vertex set V represents the objects and the edge set E represents the pairwise connections among objects. **Density**, the primary graph descriptor, plays an important role in the identification of semantically important regions of a given graph. The density of graph G is defined as $\rho(G) = \frac{|E|}{|V|}$, the ratio of the number of edges and the number of nodes[1]. Let S be a subset of nodes of G. We abuse S to denote S's induced subgraph whose vertex set is S and edge set $E(S) \subseteq E$ is the set of all edges with both end-points in S. The subgraph that maximizes the density

$$\mathrm{DS}(G) = \arg\max_{S \subseteq V(G)} \rho(S)$$

is computed (see seminal work [28] and a recent survey [27]) for various graph analytical tasks such as community detection on social networks [19], spam link farms identification on web graphs [26], experts extraction from crowdsourcing systems [30], etc..

Reporting a subgraph with the highest density can be insufficient for an application that expects multiple possible resulting subgraphs. A line of research [6, 12, 39, 42, 43, 46] diversifies densest subgraph search such that given a node (set), the densest subgraph relevant to the query node(s) can be reported[2]. However, with the density notion unchanged, algorithms in [24, 39, 46] can only get one densest subgraph S on a graph G. They perform flow-based densest subgraph search on an iteratively shrinking G to make sure that all the resulting subgraphs are disjoint. This leads to expensive computation especially on real-world graphs with millions of nodes and billions of edges. Rigid overlapping constraints can also lead to degeneration: the resulting subgraphs in [12, 39, 46] are required to be either disjoint or nested. When the graph G is connected and regular (namely, all the nodes in G are of the same degree), only one possible subgraph can be reported by [12, 39], regardless of the query node(s). Softerning the overlapping constraints of the resulting subgraphs may lead to NP-hard computations [6].

Therefore, given a query node (set), how to define the density of an arbitrary node set S is challenging for the following reasons.

(1) The densest subgraph found under the new density definition should have wide real-world applications;
(2) The density should reflect to what extent S is biased to the query node (set) while avoiding degeneration;
(3) The computation of the densest subgraph under the new definition should be efficient even on billion-scale graphs.

To address the first challenge, we make observations on local community search (see [49] as an entrance) applications.

$\underline{\mathrm{UC}_1}$**.** Consider a person x who plans to organize a social event to either establish or enhance his professional or personal connections. He may have a collection R of people who can potentially help establish the connection and a set $A \subseteq R$ of must-attend people. He poses a query to a social network by providing node sets A and R and expects a densely connected community that includes all people in A and preferably people in R. In finding the densest subgraph around A and R as a community, it is likely that including a celebrity y can increase the density of the resulting subgraph. However, if

[1] Density can also be defined as $\frac{|E|}{|V|(|V|-1)}$, which will not be discussed in this paper.

[2] Reporting the densest subgraph around a query node (set) is an instance of local community detection; however, we avoid using the phrase "local densest subgraph" because it carries a different meaning in the literature of densest subgraph search [39].

SIGMOD '22, June 12–17, 2022, Philadelphia, PA, USA.

ACM ISBN 978-1-4503-9249-5/22/06…$15.00
https://doi.org/10.1145/3514221.3517890

y is too popular to be within the reach of x, recommending y as a member of the local community could be unwise.

From the above case we find that *nodes with significantly higher centrality than the query node(s) may boost the density in the densest subgraph search while recommending them to be in the local community can be risky.* The use case below carries the same message.

$\underline{\text{UC}_2}$**.** Consider the co-purchasing network of an e-commerce where each node is a product and an edge between two nodes denotes the co-purchasing of the two items[3]. Given a set R of the items visited by a user x since the last transaction and a set $A \subseteq R$ of products that were added to x's shopping cart, the system would like to recommend the items that are most likely to be co-purchased with A given the preferred items in R. Note that a large number of items may have been co-purchased with popular items such as toilet paper by past users; however, if most of the items visited by x are in special categories, e.g., comic books of a specific series or skateboard gears, then it is unwise to let toilet paper supersede unpopular items that are highly correlated to R in the recommendation.

Based on the above use cases, we derive the problem of **anchored densest subgraph** (ADS) search. A user inputs an *anchored node set* A, $A = \emptyset$ if not specified, and a *reference node set* R such that i) $A \subseteq R$, and ii) the induced subgraph of R has at least one edge. Upon receiving a query of A and R, ADS reports the **densest** subgraph that combines **two-dimensional locality**:

Inclusion	ADS includes all the nodes in A and is biased to R[4];
Centrality	ADS considers nodes with centralities comparable to the nodes in R; in other words, nodes with significantly higher centralities are less likely to be included.

For a reference node set R, to combine the density with the locality, define the *R-subgraph density* of an arbitrary set S of nodes

$$\rho_R(S) = \frac{2|E(S)| - \sum_{v \in S \text{ and } v \notin R} d(v)}{|S|}.$$

Here $d(v)$ denotes the degree of node v in the graph G — we use node degree to reflect the node centrality in this definition. ADS reports the *supergraph* of A that maximizes the R-subgraph density

$$\arg\max_{S:\, A \subseteq S \subseteq V} \rho_R(S).$$

R-subgraph density differs from the double of traditional density, $\frac{2|E(S)|}{|S|}$, with a penalty term $\sum_{v \in S \text{ and } v \notin R} d(v)$ in the numerator. This term penalizes, for each node $v \in S$ in neither A nor R, an amount proportional to the centrality (degree) of v. In other words, a high degree node not in R incurs a high penalty. This way, ADS softly adjusts the locality of the reported subgraph with R. Figure 1 shows an example of the ADS of a query with A and R.

Apart from proposing the problem of ADS, this paper provides a **local algorithm** for the query processing of ADS. A local algorithm has complexity (both space and time) *only related to the input nodes in R* as opposed to the node set of the entire graph G. Recall that the flow-based densest subgraph search cannot, so far, break the quadratic barrier. Bearing this constraint, local computation is dedicated to bounding the size of the graph on which the network flow will be conducted. Our local computation guarantee is critical to bounding the time and space complexity: for graphs with billions of edges, as long as the query is local, i.e., the number of edges on nodes in R is small, the computation requires little resource.

[3]Our definition and solution can be extended to positively weighted undirected graphs and thus we can set the edge weight to the frequency of the co-purchase.

[4]Constraint $A \subseteq R$ makes sense since a subgraph is biased to the nodes that it includes.

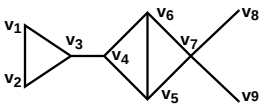


Figure 1: Graph G has 9 vertices and 11 edges. The density of G is $\rho(G) = \frac{11}{9}$. The induced subgraph of $S_{1-7} = \{v_1, v_2, \cdots, v_7\}$ has density $\rho(S_{1-7}) = \frac{9}{7}$, which is the densest subgraph of G. Let $A = \{v_1\}$ be the anchored node set and $R = \{v_1, v_3\}$ the reference node set. The anchored densest subgraph is $\text{ADS}(A, R) = \{v_1, v_2, v_3\}$ with R-subgraph density $\frac{6-2}{3} = \frac{4}{3}$ since the only node in S not in R is v_2.

Our work provides the **first** local algorithm for local community search based on the notion of density. It is worth mentioning that our work was inspired by the local community search for the notion of *conductance* (see [23, 38] as entrance); however, the density-based problem formulation and local algorithm design are non-trivial, which can be reflected by the fact that apart from conductance, no other notion has led to a local algorithm prior to our work.

The contribution of this paper is summarized as follows.

(1) **Problem.** We propose the problem of anchored densest subgraph (ADS) which is rooted in real-world applications. The definition is a concise and clean augmentation to the densest subgraph search, which entails soft overlapping constraints without requiring user-defined real-value parameters. We provide use cases of ADS and a case study in comparing the results of ADS with that of other local community search algorithms.
(2) **Solution.** We provide a local algorithm for computing the anchored densest subgraph. The algorithm is local because the complexity of the algorithm is only related to the input set A and R, irrelevant to the size of the entire graph G. This locality of the algorithm allows low latency in query processing and is friendly to concurrent queries on massive graphs.
(3) **Experiments.** Extensive experiments show our superiority: the local algorithm outperforms the global algorithm by up to **three orders of magnitudes** in both time consumption and space consumption. Our case study on the real data set of Amazon shows the effectiveness of our proposed problem and solution in local community detection.

The paper is organized as follows. Section 2 provides preliminaries and problem definitions. Section 3 proposes a global algorithm that runs in time polynomial to the size of the input graph G. Section 4 proposes a local algorithm whose time and space complexity are both bounded by a polynomial of $\sum_{v \in R \cup A} d(v)$. Section 5 discusses related works. Section 6 demonstrates empirical studies evaluating the effectiveness of our proposed problem and the efficiency of our proposed algorithms. Section 7 concludes the paper.

2 PRELIMINARIES

This section formally defines the problem, including the concepts that may have been introduced in Section 1. In this paper, we mainly consider *unweighted and undirected* graph $G = (V, E)$ while our techniques can be extended to positively *weighted* graphs. V is the set of vertices and E is the set of edges. Denote, by n and m, respectively, the number of vertices and the number of the edges in

G. Denote the edge between u and v by both (u, v) and (v, u); then, u (resp. v) is said to be adjacent to and a neighbor of v (resp. u). The set of neighbors of u in G is $N_G(u) = \{v \in V \mid (u, v) \in E\}$, and the *degree* of u in G is $d_G(u) = |N_G(u)|$. Given a vertex subset S of G, denote by ∂S the set of neighbors of vertices in S, *i.e.*, $\partial S = \{v \in V \setminus S \mid \exists u \in S \text{ s.t. } (u, v) \in E\}$, and by $E(S)$ the set of edges whose both end-points are in S, *i.e.*, $E(S) = \{(u, v) \in E \mid u, v \in S\}$. Let $\bar{S}$ be the compliment of S in V, *i.e.*, $\bar{S} = V \setminus S$. Denote by $E(S, \bar{S})$ the set of edges between S and $\bar{S}$, *i.e.*, $E(S, \bar{S}) = E \cap (S \times \bar{S})$. When the context is clear, we abbreviate $N_G(u)$ and $d_G(u)$ as $N(u)$ and $d(u)$. For an arbitrary graph g, denote by $V(g)$ and $E(g)$, respectively, the set of vertices and set of edges of g.

Given a graph $G = (V, E)$, the anchored densest subgraph problem finds, for an anchor vertex set $A \subseteq V$ and a reference vertex set $R \subseteq V$ such that $A \subseteq R$ and $E(R) \neq \emptyset$, the subgraph S^* with $A \subseteq S^*$ that maximizes the *R-subgraph density* $\rho_R(S^*)$. Denote by $\text{vol}(S) = \sum_{u \in S} d_G(u)$ the aggregated (sum) degree of vertices in S. R-subgraph density is defined below.

Definition 2.1 (R-subgraph Density). Given a graph $G = (V, E)$ and a reference vertex set $R \subseteq V$, the R-subgraph density of a non-empty vertex set $S \subseteq V$ is

$$\rho_R(S) = \frac{2|E(S)| - \sum_{v \in S \setminus R} d(v)}{|S|} = \frac{2|E(S)| - \text{vol}(S \cap \bar{R})}{|S|}.$$

Note that, the definition of R-subgraph density aims at localizing the notion of density. If S is not local to R, then $\rho_R(S)$ may be negative. Nevertheless, the maximum R-subgraph density is always positive, as it is no smaller than $\rho_R(R) = \frac{2|E(R)|}{|R|} > 0$. Also, it is worth pointing out that vertices of R are not penalized by the R-subgraph density.

Thus, given A and R with $A \subseteq R$ and $E(R) \neq \emptyset$, our problem is to find the subgraph S^* with the maximum R-subgraph density, *i.e.*,

$$S^* = \underset{S: A \subseteq S \subseteq V, S \neq \emptyset}{\arg\max} \rho_R(S) = \underset{S: A \subseteq S \subseteq V, S \neq \emptyset}{\arg\max} \frac{2|E(S)| - \text{vol}(S \cap \bar{R})}{|S|}.$$

In the following, we denote by ρ_R^* the maximum R-subgraph density, *i.e.*,

$$\rho_R^* = \max_{S: A \subseteq S \subseteq V, S \neq \emptyset} \rho_R(S).$$

Frequently used notations are summarized in Table 1.

3 A GLOBAL ALGORITHM

This section presents a baseline approach called the *global* algorithm. It reports the exact solution to anchored densest subgraph problem with running time polynomial to the size of the input graph G. The global algorithm iteratively probes, using binary search, a value α for ρ_R^*: if $\alpha < \rho_R^*$, then we increase our guess value α; otherwise, we decrease our guess value α. The key to the algorithm is to efficiently check whether $\alpha < \rho_R^*$. In the following, we first present in Section 3.1 an algorithm for efficiently checking whether $\alpha < \rho_R^*$, then discuss how to conduct the binary search in Section 3.2, and finally present the overall global algorithm in Section 3.3.

3.1 Checking Whether $\alpha < \rho_R^*$

To efficiently check whether $\alpha < \rho_R^*$, we derive an equivalent condition of $\rho_R(S) > \alpha$ for $\forall\, \alpha \in \mathbb{R}_+$ and set $S \subseteq V$ with $S \neq \emptyset$.

Table 1: Frequently used notations

Notation	Definition
$N_G(u)$	the set of neighbors of u in G: $N_G(u) = \{v \in V \mid (u, v) \in E\}$
$d_G(u)$	the degree of u in G: $d_G(u) = \lvert N_G(u)\rvert$
$d_{\max}(R)$	the maximum degree among vertices of R: $d_{\max}(R) = \max_{u \in R} d_G(u)$
∂S	the set of neighbors of vertices of S: $\partial S = \{v \in V \setminus S \mid \exists u \in S \text{ s.t. } (u, v) \in E\}$
$E(S)$	the set of edges with both end-points in S: $E(S) = \{(u, v) \in E \mid u, v \in S\}$
$\bar{S}$	the complement of S in V: $\bar{S} = V \setminus S$
$E(S, \bar{S})$	the set of edges between S and $\bar{S}$: $E(S, \bar{S}) = E \cap (S \times \bar{S})$
$\text{vol}(S)$	the sum of the degrees of vertices of S: $\text{vol}(S) = \sum_{u \in S} d_G(S)$
$\rho_R(S)$	R-subgraph density of S: $\rho_R(S) = \frac{2\lvert E(S)\rvert - \text{vol}(S \cap R)}{\lvert S\rvert}$
ρ_R^*	the maximum R-subgraph density: $\max_{S: A \subseteq S \subseteq V, S \neq \emptyset} \rho_R(S)$
G_α	the augmented graph of G for $\alpha \in \mathbb{R}_+$
$(\{s\} \cup S, \{t\} \cup \bar{S})$	the s–t cut in G_α that consists of all edges between $\{s\} \cup S$ and $\{t\} \cup \bar{S}$

Lemma 3.1. *For $\forall \alpha \in \mathbb{R}_+$ and $\forall S \subseteq V$ with $S \neq \emptyset$, it holds that*

$$\rho_R(S) > \alpha \iff \text{vol}(\bar{S} \cap R) + |E(S, \bar{S})| + \alpha|S| < \text{vol}(R).$$

Similarly, $\rho_R(S) = \alpha \iff \text{vol}(\bar{S} \cap R) + |E(S, \bar{S})| + \alpha|S| = \text{vol}(R).$

Proof. As $S \neq \emptyset$ and $2|E(S)| = \text{vol}(S) - |E(S, \bar{S})|$, we have

$$\begin{aligned}
\rho_R(S) > \alpha &\iff 2|E(S)| - \text{vol}(S \cap \bar{R}) > \alpha|S| \\
&\iff \text{vol}(S) - |E(S, \bar{S})| - \text{vol}(S \cap \bar{R}) > \alpha|S| \\
&\iff \text{vol}(S \cap R) - |E(S, \bar{S})| > \alpha|S| \\
&\iff \text{vol}(R) - \text{vol}(\bar{S} \cap R) - |E(S, \bar{S})| > \alpha|S| \\
&\iff \text{vol}(R) > \text{vol}(\bar{S} \cap R) + |E(S, \bar{S})| + \alpha|S|
\end{aligned}$$

The case of equality can be obtained by replacing ">" with "=" in the above inference. □

Note that, the above lemma holds for any non-empty $S \subseteq V$, and thus it also holds for those S that contain the anchor vertex set A. According to the definition of ρ_R^* (in Section 2), for any fixed α, we have $\alpha < \rho_R^*$ if and only if there exists a non-empty vertex subset $S \subseteq V$ satisfying $A \subseteq S$ and $\alpha < \rho_R(S)$. Thus, from Lemma 3.1 and the definition of ρ_R^*, we have the following lemma.

Lemma 3.2. *For any fixed α, $\alpha < \rho_R^*$ if and only if there exists a non-empty set $S \subseteq V$ satisfying $A \subseteq S$ and* $\text{vol}(\bar{S} \cap R) + |E(S, \bar{S})| + \alpha|S| < \text{vol}(R)$.

Similarly, it also holds that $\alpha = \rho_R^*$ if and only if

(1) there exists a non-empty vertex subset S with $A \subseteq S \subseteq V$ satisfying $\text{vol}(\bar{S} \cap R) + |E(S, \bar{S})| + \alpha|S| = \text{vol}(R)$, and
(2) every non-empty vertex subset S' with $A \subseteq S' \subseteq V$ satisfies $\text{vol}(\bar{S'} \cap R) + |E(S', \bar{S'})| + \alpha|S'| \geq \text{vol}(R)$.

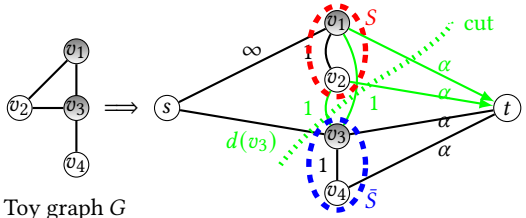
Here, condition (1) is equivalent to $\rho_R^* \geq \alpha$, and condition (2) is equivalent to $\rho_R^* \leq \alpha$.

For a specific α, to efficiently find such an S satisfying $A \subseteq S$ and $\text{vol}(\bar{S} \cap R) + |E(S, \bar{S})| + \alpha|S| < \text{vol}(R)$ (this corresponds to the case $\alpha < \rho_R^*$) or prove that there is no such an S (this corresponds to the case $\alpha \geq \rho_R^*$), we construct an *edge-weighted* and *undirected* graph,

called the *augmented graph* and denoted as G_α, by augmenting $G = (V, E)$ as follows.

- Add a source vertex s and a sink vertex t to $V(G_\alpha)$, where $s, t \notin V$, *i.e.*, $V(G_\alpha) = V \cup \{s, t\}$.
- Each edge of E bears a weight of 1.
- Add an edge between the source vertex s and each vertex $u \in A$ with weight $+\infty$.
- Add an edge between the source vertex s and each vertex $u \in R \setminus A$ with weight $d_G(u)$.
- Add an edge between each vertex $v \in V$ and the sink vertex t with weight α.

An example G_α is shown in Figure 2, where $A = \{v_1\}$ and $R = \{v_1, v_3\}$. There are two things worth emphasizing for the augmented graph. Firstly, s is connected only to vertices of R. Secondly, the edges between s and vertices of A all have infinity weight, *e.g.*, see the edge between s and v_1 in Figure 2.



Toy graph G
$A = \{v_1\}, R = \{v_1, v_3\}$ Augmented Graph G_α

Figure 2: Augmented graph G_α (vertices of R are shadowed)

The main benefit of constructing G_α is that it bridges the R-subgraph density of a vertex subset in G to the value of an s–t cut in G_α. For any $S \subseteq V$, we let $(\{s\} \cup S, \{t\} \cup \bar{S})$ denote the s–t cut in G_α that consists of all the cross edges between vertices of $\{s\} \cup S$ and vertices of $\{t\} \cup \bar{S}$; recall that G_α is undirected, and thus a cut consists of undirected edges. We denote the total weight of all edges in $(\{s\} \cup S, \{t\} \cup \bar{S})$ by $\text{cut}(\{s\} \cup S, \{t\} \cup \bar{S})$; this is also known as the value of the cut $(\{s\} \cup S, \{t\} \cup \bar{S})$. We have the following lemma regarding the value of an s–t cut in G_α.

Lemma 3.3. *For any vertex subset $S \subseteq V$ that contains the anchor vertex set A, the value of the cut $(\{s\} \cup S, \{t\} \cup \bar{S})$ in G_α is equal to* $\text{vol}(\bar{S} \cap R) + |E(S, \bar{S})| + \alpha|S|$.

Proof. The edges of cut $(\{s\} \cup S, \{t\} \cup \bar{S})$ can be divided into three parts, as demonstrated in Figure 2: (1) the edges between s and $\bar{S}$ whose total weight is $\text{vol}(\bar{S} \cap R)$ as there is no edge between s and $\bar{R}$, (2) the edges between S and $\bar{S}$ whose total weight is $|E(S, \bar{S})|$, and (3) the edges between S and t whose total weight is $\alpha|S|$. Note that, there is no edge between s and t. □

Example 3.4. The five green edges in G_α in Figure 2 are the edges in the cut $(\{s, v_1, v_2\}, \{t, v_3, v_4\})$, and the value of the cut is $d(v_3) + 2 + 2\alpha = \text{vol}(\bar{S} \cap R) + |E(S, \bar{S})| + \alpha|S|$ where $S = \{v_1, v_2\}$.

Combining Lemma 3.2 and Lemma 3.3, we have that $\alpha < \rho_R^*$ if and only if there exists a non-empty vertex subset $S \subseteq V$ satisfying $A \subseteq S$ and the value of the cut $(\{s\} \cup S, \{t\} \cup \bar{S})$ in the augmented graph G_α is less than $\text{vol}(R)$. Moreover, the way that the edge weights are assigned in G_α makes it trivial to enforce the condition $A \subseteq S$ via minimum s–t cut — the s–t cut with the lowest total weight. That is, for any minimum s–t cut $(\{s\} \cup S, \{t\} \cup \bar{S})$, we have $A \subseteq S$; this is because all edges between s and A have infinity weight. Combining Lemmas 3.2 and 3.3, we have the following lemma that compares α with ρ_R^* via the minimum s–t cut in G_α.

Lemma 3.5. *$\alpha < \rho_R^*$ if and only if the minimum s–t cut value of the augmented graph G_α is strictly smaller than* $\text{vol}(R)$.

As we will only utilize minimum s–t cut of G_α, in the following discussions, we will not explicitly mention the fact that $A \subseteq S$ **constantly holds on the minimum cut**.

Lemma 3.5 only utilizes the value of the minimum s–t cut of G_α. We can also characterize the actual minimum s–t cut of G_α for the special case $\alpha = \rho_R^*$ by the lemma below.

Lemma 3.6. *For $\alpha = \rho_R^*$, G_α has at least one minimum s–t cut $(\{s\} \cup S, \{t\} \cup \bar{S})$ with $S \neq \emptyset$, and moreover, for every such minimum s–t cut $(\{s\} \cup S, \{t\} \cup \bar{S})$ with $S \neq \emptyset$, S is an anchored densest subgraph* (i.e., $\rho_R(S) = \rho_R^*$).

Proof. From the above discussions, we know that there must exist a non-empty vertex subset $S \subseteq V$ satisfying $\text{vol}(\bar{S} \cap R) + \text{cut}(S) + \alpha|S| = \text{vol}(R)$; that is, the value of the s–t cut $(\{s\} \cup S, \{t\} \cup \bar{S})$ is $\text{vol}(R)$. Lemma 3.5 implies that the minimum s–t cut of G_α is of value at least $\text{vol}(R)$ as $\alpha = \rho_R^*$. Thus, G_α has at least one minimum s–t cut $(\{s\} \cup S, \{t\} \cup \bar{S})$ with $S \neq \emptyset$. Next, let's consider any minimum s–t cut $(\{s\} \cup S, \{t\} \cup \bar{S})$ with $S \neq \emptyset$. The value of the cut must be $\text{vol}(R)$. Thus, $\text{vol}(\bar{S} \cap R) + |E(S, \bar{S})| + \alpha|S| = \text{vol}(R)$ by following Lemma 3.3, and then, $\rho_R(S) = \alpha = \rho_R^*$ by following Lemma 3.1. □

3.2 Binary Search

Following Section 3.1, we conduct a binary search on ρ_R^* to first find the anchored densest subgraph S^*, and then obtain the exact value of ρ_R^*; note that, the binary search itself will not directly compute the exact value of ρ_R^*, as float numbers cannot be represented exactly in a computer. There are three more questions to be answered in this subsection: what is the initial range of the binary search, when to stop the binary search, and for what values of α that are not exactly ρ_R^* we are able to find the anchored densest subgraph?

Recall that, we assumed $A \subseteq R$ and $E(R) \neq \emptyset$. We first prove that the maximum R-subgraph density ρ_R^* is in the range between $\frac{2}{\min\{|A|+2, |R|\}}$ and $d_{\max}(R)$, where $d_{\max}(R)$ denotes the maximum degree among vertices of R (*i.e.*, $d_{\max}(R) = \max_{u \in R} d_G(u)$).

Lemma 3.7. $\frac{2}{\min\{|A|+2, |R|\}} \leq \rho_R^* \leq d_{\max}(R)$.

Proof. Firstly, for any two vertices $u, v \in R$ that are connected by an edge, we have $\rho_R(A \cup \{u, v\}) \geq \frac{2}{\min\{|A|+2, |R|\}}$. Consequently,

$$\rho_R^* = \max_{S: A \subseteq S \subseteq V, S \neq \emptyset} \rho_R(S) \geq \rho_R(A \cup \{u, v\}) \geq \frac{2}{\min\{|A|+2, |R|\}}.$$

Secondly, from the proof of Lemma 3.1, we have that for any non-empty vertex subset $S \subseteq V$,

$$\rho_R(S) = \frac{2|E(S)| - \text{vol}(S \cap \bar{R})}{|S|} = \frac{\text{vol}(S \cap R) - |E(S, \bar{S})|}{|S|} \leq \frac{\text{vol}(S \cap R)}{|S|} \leq \frac{|S \cap R| \cdot d_{\max}(R)}{|S|} \leq d_{\max}(R)$$

Thus, $\rho_R^* \leq d_{\max}(R)$. □

Next, we prove that the gap between any two distinct R-subgraph densities is at least $\frac{1}{n(n-1)}$.

Lemma 3.8. *For any two non-empty vertex subsets S and S' that have different R-subgraph densities (*i.e.*, $\rho_R(S) \neq \rho_R(S')$), it holds that $|\rho_R(S) - \rho_R(S')| \geq \frac{1}{n(n-1)}$.*

Proof. From the definition of R-subgraph density, we have

$$|\rho_R(S) - \rho_R(S')| = \left|\frac{2|E(S)|-\text{vol}(S\cap\bar{R})}{|S|} - \frac{2|E(S')|-\text{vol}(S'\cap\bar{R})}{|S'|}\right|$$
$$= \frac{||S'|(2|E(S)|-\text{vol}(S\cap\bar{R}))-|S|(2|E(S')|-\text{vol}(S'\cap\bar{R}))|}{|S|\cdot|S'|} \geq \frac{1}{n(n-1)}$$

The last inequality follows from the facts that $||S'|(2|E(S)|-\text{vol}(S\cap\bar{R})) - |S|(2|E(S')| - \text{vol}(S'\cap\bar{R}))| \geq 1$ and $|S|\cdot|S'| \leq n(n-1)$. □

Thus, we can terminate the binary search once the range is smaller than $\frac{1}{n(n-1)}$. Moreover, we have the following corollary.

Corollary 3.9. *For any $\alpha \in [\rho_R^* - \frac{1}{n(n-1)}, \rho_R^*)$, any minimum s–t cut $(\{s\} \cup S, \{t\} \cup \bar{S})$ of G_α corresponds to an anchored densest subgraph,* i.e.*, S is an anchored densest subgraph.*

Proof. Firstly, from Lemma 3.5 we know that as $\alpha < \rho_R^*$, the cut $(\{s\} \cup S, \{t\} \cup \bar{S})$ is of value strictly smaller than $\text{vol}(R)$ and is thus not trivial (*i.e.*, $A \subseteq S$ and $S \neq \emptyset$). Secondly, from Lemma 3.3 we know that $\text{vol}(\bar{S} \cap R) + |E(S, \bar{S})| + \alpha|S| < \text{vol}(R)$. Thirdly, from Lemma 3.1 we know that $\rho_R(S) > \alpha$. Lastly, as $\alpha \geq \rho_R^* - \frac{1}{n(n-1)}$, we know from Lemma 3.8 that $\rho_R(S)$ must be exactly ρ_R^* and thus S is an anchored densest subgraph. □

3.3 Putting All Together

Based on the ideas in Sections 3.1 and 3.2, our global algorithm for the anchored densest subgraph search problem is shown in Algorithm 1. We first initialize the lower bound lb and the upper bound ub of ρ_R^* as 0 and $d_{\max}(R)$ (Line 2), respectively, by following Lemma 3.7; note that, although we proved in Lemma 3.7 that $\rho_R^* > 0$, we set $lb = 0$ for the sake of easily handling degenerate cases. Then, we conduct binary search to narrow down the range $[lb, ub]$. Specifically, let $\alpha = \frac{lb+ub}{2}$ be the middle point of $[lb, ub]$ (Line 4). We check whether $\alpha < \rho_R^*$ or not. To do that, we construct the augmented graph G_α (Line 5), and compute a minimum s–t cut for G_α (Line 6). If the minimum cut value is at least $\text{vol}(R)$, then we know from Lemma 3.5 that $\alpha \geq \rho_R^*$ and we thus narrow down the range to $[lb, \alpha]$ (Line 7). Otherwise, $\alpha < \rho_R^*$ and we narrow down the range to $[\alpha, ub]$ (Line 8). The last non-trivial cut S is returned as the anchored densest subgraph (Line 9).

It is easy to verify by induction that Algorithm 1 maintains the invariant that $lb < \rho_R^* \leq ub$. As the algorithm terminates only when $ub - lb \leq \frac{1}{n(n-1)}$ (Line 3), the following holds when the algorithm terminates:

$$lb \geq ub - \frac{1}{n(n-1)} \geq \rho_R^* - \frac{1}{n(n-1)} > 0$$

From the above inequalities we conclude two things: (1) lb has changed from its initialization which is 0, and (2) the last change of lb is by $\alpha \in [\rho_R^* - \frac{1}{n(n-1)}, \rho_R^*)$. Thus, following Corollary 3.9, the last non-trivial cut is the anchored densest subgraph.

Algorithm 1 generates $O(\log n)$ instances of minimum s–t cut computation (Line 6). Thus, its time complexity is $O(\log n)$ times the best time complexity of minimum s–t cut computation algorithms; note that, any minimum s–t cut computation algorithm can be invoked here. By leveraging the push-relabel technique [25] which does not need to conduct binary search, the time complexity of Algorithm 1 is $O(nm \log \frac{n^2}{m})$.

Algorithm 1: Global(G, A, R)

Input: A graph $G = (V, E)$, an anchor vertex set $A \subseteq V$ and a reference vertex set $R \subseteq V$ satisfying $A \subseteq R$ and $E(R) \neq \emptyset$
Output: An anchored densest subgraph S^*

1 $S^* \leftarrow \emptyset$;
2 $lb \leftarrow 0$; $ub \leftarrow d_{\max}(R)$;
3 **while** $ub - lb > \frac{1}{n(n-1)}$ **do**
4 $\alpha \leftarrow \frac{lb+ub}{2}$;
5 $G_\alpha \leftarrow$ construct augmented graph;
6 Compute a minimum s–t cut of G_α, and denote it by $(\{s\} \cup S, \{t\} \cup \bar{S})$;
7 **if** $\text{cut}(\{s\} \cup S, \{t\} \cup \bar{S}) \geq \text{vol}(R)$ **then** $ub \leftarrow \alpha$;
8 **else** $lb \leftarrow \alpha$; $S^* \leftarrow S$;
9 **return** S^*;

4 A LOCAL ALGORITHM

Although the global algorithm proposed in Algorithm 1 runs in polynomial time, it is not scalable for massive graphs. This section proposes a local algorithm for exact anchored densest subgraph computation. Its running time is bounded by a polynomial of $\text{vol}(R)$ and is independent of the size of the graph G. We first propose a local algorithm to compute a minimum s–t cut in G_α in time polynomial to $\text{vol}(R)$ in Section 4.1, and then bound the number of minimum s–t cut instances by logarithm of $\text{vol}(R)$ in Section 4.2.

4.1 Locally Computing Minimum s–t Cut in G_α

In this subsection, we adopt maximum flow computation algorithms for computing the minimum s–t cut; different from Algorithm 1, we look into the maximum flow algorithms such that we can design a strategy to compute the maximum flow from s to t locally around R without visiting (as well as constructing) the entire augmented graph G_α. That is, we aim to compute a maximum flow from s to t in G_α in time independent of the size of G_α.

We first review the concepts of *flow* and *residual graph*. For maximum flow computation, we will consider the augmented graph G_α as a *directed* graph; specifically, each undirected edge is represented as two directed edges, one in each direction. A directed edge from u to v is denoted by $\langle u, v\rangle$, and is different from the directed edge $\langle v, u\rangle$ from v to u. Each edge $e \in E(G_\alpha)$ has a capacity, denoted $c(e)$, which is equal to its weight as defined in Section 3.1.

Definition 4.1 (Flow). An s–t flow of G_α is a mapping f from each edge $e \in E(G_\alpha)$ to a real value $f(e) \in \mathbb{R}$ that satisfies the following three properties:

Capacity: $f(e) \leq c(e)$ for each edge $e \in E(G_\alpha)$;
Antisymmetry: $f(\langle u, v\rangle) = -f(\langle v, u\rangle)$ for $\forall\langle u, v\rangle \in E(G_\alpha)$;
Conservation: $\sum_{e\in E(G_\alpha) \text{ into } u} f(e) = \sum_{e\in E(G_\alpha) \text{ out } u} f(e) = 0$ for each vertex $u \in V(G_\alpha) \setminus \{s, t\}$.

The *value* of the flow f is defined as

$$val(f) = \sum_{e\in E(G_\alpha) \text{ out } s} f(e) = \sum_{e\in E(G_\alpha) \text{ into } t} f(e)$$

A maximum s–t flow of G_α is the s–t flow f that maximizes $val(f)$. The famous *max-flow min-cut theorem* states that the value of a minimum s–t cut is equal to the value of a maximum s–t flow [16]. Moreover, a minimum s–t cut can be obtained from a maximum s–t flow [16], based on the concept of residual graph. In the following, for simplicity, we will refer to an s–t flow as a flow.

Definition 4.2 (Residual Graph). Given any flow f of G_α, the residual graph, denoted by $G_\alpha \oplus f$, has the same set of vertices and edges as G_α. The capacity of an edge e in $G_\alpha \oplus f$ is denoted as $c_f(e)$ and is defined as $c(e) - f(e)$. An edge is called *saturated* if $c_f(e) = 0$ or equivalently $f(e) = c(e)$. A flow f is maximum if and only if there is no path from s to t in $G_\alpha \oplus f$ by visiting only non-saturated edges.

Our main idea is to utilize the fact that s is adjacent to only vertices of R in G_α as shown in Figure 2. Thus, it is expected that the maximum flow only visits vertices around R in G_α. Obviously, it is not affordable to materialize the entire graph G_α, as otherwise the running time would depend on the size of G_α and thus also the size of G. Motivated by this, we propose to compute maximum flows from s to t in a series of iteratively growing subgraphs of G_α. For any vertex subset $S \subset V$, recall that ∂S denotes the set of neighbors of S that are not in S, *i.e.*, $\partial S = (\cup_{u \in S} N_G(u)) \setminus S$. Then, for any vertex subset $B \subseteq V \setminus R$, we extract a subgraph of G_α for vertices $\{s, t\} \cup (R \cup B) \cup \partial(R \cup B)$, and denote the subgraph as $g_{\alpha,B}$. Specifically, $g_{\alpha,B}$ consists of

- vertices $\{s, t\} \cup R \cup B \cup \partial(R \cup B)$,
- all adjacent edges of $R \cup B$ in G_α,
- all edges between $\partial(R \cup B)$ and t,

A schematic example of $g_{\alpha,B}$ is shown in Figure 3, where the entire graph is G_α. Note that

- s is adjacent only to vertices of R,
- there is no edge between $\{s\} \cup R \cup B$ and $V \setminus (R \cup B \cup \partial(R \cup B))$,
- edges between vertices of $\partial(R \cup B)$ are not included in $g_{\alpha,B}$.

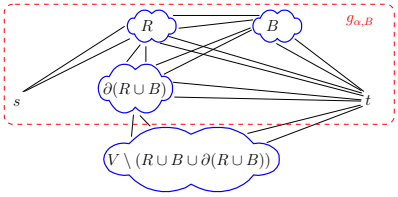


Figure 3: Illustration of $g_{\alpha,B}$ in G_α

The main result is that if none of the edges from $\partial(R \cup B)$ to t is saturated in a maximum flow f from s to t in $g_{\alpha,B}$, then this flow f is also a maximum flow in G_α. Note that, here we assume $f(e) = 0$ for all $e \in E(G_\alpha) \setminus E(g_{\alpha,B})$, as they are not defined in $g_{\alpha,B}$.

Lemma 4.3. *If none of the edges from $\partial(R \cup B)$ to t is saturated in a maximum flow f from s to t in $g_{\alpha,B}$, then this flow f is also a maximum flow in G_α.*

Proof. This can be seen by contradiction. Suppose f is not a maximum flow in G_α. As f is a flow in G_α but not maximum, there must be a simple path P from s to t consisting of only non-saturated edges in the residual graph $G_\alpha \oplus f$. Furthermore, the path P must have a vertex not in $R \cup B \cup \{s, t\}$, as otherwise f is not a maximum flow in $g_{\alpha,B}$. Let u be the first vertex of P that is not in $R \cup B \cup \{s, t\}$. Then, u must be in $\partial(R \cup B)$, as there is no edge between $\{s\} \cup R \cup B$ and $V \setminus (R \cup B \cup \partial(R \cup B))$. As the edge from u to t is not saturated in $g_{\alpha,B} \oplus f$ (by the assumption of the lemma), there must also exist a path consisting of only non-saturated edges in the residual graph $g_{\alpha,B} \oplus f$; this contradicts that f is a maximum flow in $g_{\alpha,B}$. □

Based on the above ideas, we iteratively grow B to find the appropriate $g_{\alpha,B}$, until the condition in Lemma 4.3 is satisfied. Specifically, initially B is $\emptyset$, and the flow f is 0 for all edges. Then, we iteratively do the following

(1) Compute a maximum flow $\hat{f}$ in the residual graph $g_{\alpha,B} \oplus f$.
(2) If there is no flow from s to t in $g_{\alpha,B} \oplus f$, then we stop.
(3) Otherwise, we add $\hat{f}$ to f, and let B be the set of vertices of $\bar{R}$ whose edges to t are saturated in $g_{\alpha,B} \oplus f$. Then, we continue for the next iteration.

Note that, we always compute maximum flow in residual graphs $g_{\alpha,B} \oplus f$. Thus, once the edge from a vertex $u \in \bar{R}$ to t is saturated in a round of maximum flow computation, it will remain saturated in all following rounds of maximum flow computations; here, we assume that we never push flows along the edges going out of t, which is the case for all maximum flow algorithms. Thus, let B_i be the value of B after the i-th iteration, then we have $\emptyset = B_0 \subset B_1 \subset B_2 \subset \cdots$, except that the last two values of B may be the same.

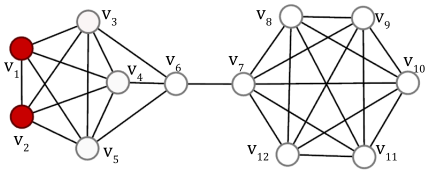


Figure 4: An example graph for our local algorithm

Example 4.4. Consider the graph in Figure 4 with $A = \emptyset$, $R = \{v_1, v_2\}$ and $\alpha = 2$. For the first iteration, $B_0 = \emptyset$, $\partial(R \cup B_0) = \{v_3, v_4, v_5\}$, the value of the maximum flow is 8 and all the three edges between $\partial(R \cup B_0)$ and t are saturated. Thus, we update $B_1 = \{v_3, v_4, v_5\}$. In the second iteration, $\partial(R \cup B_1) = \{v_6\}$, the value of the maximum flow remains 8 and the edge between v_6 and t is not saturated. Thus, $B_2 = B_1$ and the algorithm terminates.

Theoretical Analysis. Let B^* be the value of B when the algorithm terminates. Then, the size of each of the subgraphs $g_{\alpha,B_0}, g_{\alpha,B_1}, \ldots$ is bounded by the size of g_{α,B^*}. We first bound $|B^*|$.

Lemma 4.5. $|B^*| \leq \frac{\mathrm{vol}(R)}{\alpha} + |R|$.

Proof. Let f^* be the maximum flow when the algorithm terminates. Firstly, for each edge from B^* to t, its capacity is α and it is saturated in f^* according to the algorithm; thus, $val(f^*) \geq \alpha|B^*|$. Secondly, the value of the trivial cut $(\{s\} \cup R, \{t\} \cup \bar{R})$ is at most $\mathrm{vol}(R) + \alpha|R|$; thus, $val(f^*) \leq \mathrm{vol}(R) + \alpha|R|$ according to the max-flow min-cut theorem. Therefore, $\alpha|B^*| \leq val(f^*) \leq \mathrm{vol}(R) + \alpha|R|$ and the lemma follows. □

We will show later that α is no less than the lower bound of ρ_R^* during the execution of our algorithm. Recall that we proved in Lemma 3.7 that ρ_R^* is at least $\frac{2}{\min\{|A|+2, |R|\}}$ which equals 1 when

$A = \emptyset$. It is also intuitive that the result for a query is not quite useful if ρ_R^* is too small, *e.g.*, smaller than 1. Thus, for the sake of presentation simplicity, we *assume* $\rho_R^* \geq 1$ *and thus* $\alpha \geq 1$ for our theoretical analysis; note that, the locality property of our algorithm still holds, but with a different polynomial degree, if $\rho_R^* < 1$. Thus, $|B^*| \leq \text{vol}(R) + |R|$. To bound the number of edges in g_{α,B^*}, we further need the following lemma.

Lemma 4.6. *For any vertex u that is in an anchored densest subgraph, it must satisfy $d_G(u) < \text{vol}(R)$.*

Proof. Let u be any vertex in an anchored densest subgraph S. From Section 3.1, we know that $(\{s\} \cup S, \{t\} \cup \bar{S})$ must be a minimum s–t cut of G_α for $\alpha = \rho_R^* \geq 1$, and moreover, the value of the cut $(\{s\} \cup S, \{t\} \cup \bar{S})$ is $\text{vol}(R)$. Thus, we have

$$\begin{aligned}\text{vol}(R) &= \text{vol}(\bar{S} \cap R) + |E(S,\bar{S})| + \alpha|S| \\ &\geq |E(S,\bar{S})| + |S| \quad \triangleright \text{ as } \text{vol}(\bar{S} \cap R) \geq 0 \text{ and } \alpha \geq 1 \\ &\geq |E(u,\bar{S})| + |S| = d_G(u) - |E(u,S)| + |S| \geq d_G(u) + 1\end{aligned}$$

Consequently, $d_G(u) + 1 \leq \text{vol}(R)$. □

Following Lemma 4.6, in each $g_{\alpha,B}$ we can contract all vertices — whose degrees in G are at least $\text{vol}(R)$ — with the sink vertex t into a single (super) sink vertex, without affecting the minimum cut computation. Thus, we can assume that all vertices in G have degree smaller than $\text{vol}(R)$. Thus, the number of vertices in g_{α,B^*} is

$$\begin{aligned}|V(g_{\alpha,B^*})| &= 2 + |R| + |B^*| + |\partial(R \cup B^*)| \\ &\leq 2 + |R| + |B^*| + \text{vol}(R) + \sum_{u \in B^*} d_G(u) \leq 2 + 2\text{vol}(R) + 2\text{vol}^2(R).\end{aligned}$$

The last inequality follows from the facts that (1) $|R| \leq \text{vol}(R)$, and (2) $|B^*| + \sum_{u \in B^*} d_G(u) \leq |B^*|\text{vol}(R) \leq 2\text{vol}^2(R)$; we assume there is no isolated vertex in G, as they can be simply removed from G. Similarly, the number of edges of g_{α,B^*} is

$$\begin{aligned}|E(g_{\alpha,B^*})| &= 2\big(|E(R)| + |E(R, \{s,t\} \cup B^* \cup \partial(R \cup B^*))| + |E(B^*)| + \\ &\qquad |E(B^*, \{t\} \cup \partial(R \cup B^*))| + |E(\partial(R \cup B^*), \{t\})|\big) \\ &\leq 2\big(2|R| + \text{vol}(R) + |B^*| + \textstyle\sum_{u \in B^*} d_G(u) + |\partial(R \cup B^*)|\big) \\ &\leq 2\big(4\text{vol}(R) + 4\text{vol}^2(R)\big).\end{aligned}$$

The above discussions derive Lemma 4.7, bounding the size of g_{α,B^*}.

Lemma 4.7. *The number of vertices in g_{α,B^*} is $O(\text{vol}^2(R))$, and the number of edges in g_{α,B^*} is $O(\text{vol}^2(R))$.*

Thus, in each iteration (*i.e.*, for a specific B), computing a maximum flow $\hat{f}$ in the residual graph $g_{\alpha,B} \oplus f$ takes time polynomial to $\text{vol}(R)$. Furthermore, as the cardinality of B strictly increases after each iteration, the number of iterations is at most $|B^*| \leq \text{vol}(R) + |R|$. Consequently, we have the following theorem regarding the time complexity of computing a maximum flow in G_α.

Theorem 4.8. *The total time complexity of computing a maximum flow in G_α is bounded by a polynomial of $\text{vol}(R)$, and is independent of the size of G.*

Proof. This directly follows from the above discussions that bound the number of vertices and the number of edges in g_{α,B^*}. □

Note that, our analysis here is loose, as our main purpose is to show that the time complexity is independent of the size of G.

Algorithm 2: Local(G, A, R)

Input: A graph $G = (V, E)$, an anchor vertex set $A \subseteq V$ and a reference vertex set $R \subseteq V$ satisfying $A \subseteq R$ and $E(R) \neq \emptyset$
Output: An anchored densest subgraph S^*

1 $S^* \leftarrow$ {any two adjacent vertices of R};
2 $lb \leftarrow 0$; $ub \leftarrow d_{\max}(R)$;
3 **while** $ub - lb \geq \frac{1}{(3\text{vol}(R))^2}$ **do**
4 $\alpha \leftarrow \max\{\frac{lb+ub}{2}, 1\}$;
5 Compute a minimum s–t cut of G_α using the local algorithm described in Section 4.1, and denote it by $(\{s\} \cup S, \{t\} \cup \bar{S})$;
6 **if** $\text{cut}(\{s\} \cup S, \{t\} \cup \bar{S}) \geq \text{vol}(R)$ **then** $ub \leftarrow \alpha$;
7 **else** $lb \leftarrow \alpha$; $S^* \leftarrow S$;
8 **return** S^*;

4.2 Bounding the Binary Search

Section 4.1 demonstrates that we can compute a minimum s–t cut of G_α locally around R without visiting the entire graph G_α. Nevertheless, Lemma 3.8 suggests that the number of graphs G_α that we need to work on still depends on n (*i.e.*, $O(\log n)$). In this subsection, we prove that we can also bound the number of graphs G_α by logarithm of $\text{vol}(R)$.

The main idea is that we are not interested in all of the vertex subsets as proved in Lemma 3.8, instead we are only interested in the vertex subsets S that are obtained from the minimum s–t cut $(\{s\} \cup S, \{t\} \cup \bar{S})$ of G_α for $\alpha \in [1, d_{\max}(R)]$. Thus, we prove the following lemma.

Lemma 4.9. *For any two non-empty vertex subsets S and S' that have different R-subgraph densities and are obtained from the minimum s–t cuts of different augmented graphs, it holds that $|\rho_R(S) - \rho_R(S')| \geq \frac{1}{(3\text{vol}(R))^2}$.*

Proof. The general idea of the proof is the same as that in Lemma 3.8, except that we now have $|S| \leq |R| + |B^*| \leq 3\text{vol}(R)$ and similarly $|S'| \leq 3\text{vol}(R)$. This is because S in the minimum s–t cut can only be a subset of $R \cup B^*$; that is, it cannot include any vertex in $\partial(R \cup B^*)$, as none of the edges from $\partial(R \cup B^*)$ to t is saturated in a maximum flow from s to t in G_α. □

Following Lemma 4.9, we can terminate the binary search, once the range is less than $\frac{1}{(3\text{vol}(R))^2}$.

4.3 Putting All Together

Based on the ideas in Sections 4.1 and 4.2, we are now ready to present our local algorithm for anchored densest subgraph computation. The pseudocode of our local algorithm, denoted Local, is shown in Algorithm 2. It is mostly similar to Algorithm 1 but with the following differences. Firstly, we compute the minimum s–t cut in G_α using a local algorithm as discussed in Section 4.1 (Line 5). Secondly, we only work on augmented graphs G_α with $\alpha \geq 1$ (Line 4), by initializing S^* with any two adjacent vertices of R. Thus, the R-subgraph density of the initial S^* is 1, despite that S^* may not contain A. This is because we are only interested in subgraphs whose R-subgraph densities are at least 1. Thirdly, we change the stop condition at Line 3 by following Lemma 4.9. Note that, in implementation, we change the stop condition to be the same as that in Algorithm 1 if $(3\text{vol}(R))^2 > n(n-1)$.

The discussions in Sections 4.1 and 4.2 show that the time complexity of Algorithm 2 is bounded by a polynomial of vol(R), independent to the size of G. Thus, Algorithm 2 is a strongly local algorithm for computing the anchored densest subgraph.

5 RELATED WORKS

Densest Subgraph. Densest subgraph problem has been extensively studied in the literature (see survey [33]). The densest subgraph can be exactly computed in $O(nm \log \frac{n^2}{m})$ time with parametric maximum flow [25, 28], its 2-factor approximation can be computed in linear time [14]. The collection of all the densest subgraphs can be efficiently retrieved from an index structure [12]. Dense subgraph identification has also been studied on streaming and dynamic graphs [5, 8, 20].

Finding multiple dense subgraphs has also been studied [6, 18, 39, 44, 47]. For example, computing all *locally* densest subgraphs, which form a nested structure, is studied in [18, 44] — the nested subgraphs have different densities, called locally densest subgraphs, and only the inner-most subgraph is globally densest. The problem of computing top-k dense subgraphs with limited overlap is studied in [6], which iteratively computes minimal densest subgraphs and removes a certain proportion of its nodes from the input graph.

Densest subgraph definition has been generalized to higher order. The density of a subgraph in [45] is defined as the fraction of the number of h-cliques (h is a parameter) over the number of nodes in the subgraph, which degenerates to the average degree based density when $h = 2$. Fang et al. [22] further replaces the h-clique of the density defined in [45] with an arbitrary pattern graph. Sun et al. [41] improve the efficiency of finding k-clique densest subgraph by combining the idea of [18] with new insights. Dense subgraph identification based on other notions of density is surveyed in [13].

On directed graph, the densest subgraph search was formulated in [31]. Apart from an exact algorithm based on $O(n^2)$ maximum flow computations [31], Ma et al. [36] proposed an algorithm to compute a 2-approximation of the subgraph in $O(m^{3/2})$ time.

Density, together with other metrics [11] such as modularity, conductance, normalized cut, etc., serves as the optimization objectives / measurements in community detection [4].

Local Community Detection. Local community detection has been studied in literature which bisects based on whether a query (node or node set) is required for the detection. Without a query (see [4]), the locality is reflected in narrowing, in computing the correlation between a community C to the graph, the context from the entire graph to a portion of the graph that is more important to C, e.g., the *local modularity* proposed by Muff et al. [37].

Given a query (node or node set), existing local community detection has three categories, metric optimization, random walk and motif-based search. Metric optimization [15, 38, 48, 49] has two lines of research, one line uses global metric and starts the search from the query node(s). For example, Greedy-L [15] searches, from the seed node, a community that optimizes a newly proposed metric L while two algorithms (Emc, PGDc) proposed in [48] search the graph from the query node to optimize the σ-conductance, a conductance measure that uses σ to control the size of the preferred community. Since the measure is still global, the complexity of the search is polynomial to the graph size. The other line of research optimizes the measurement that is defined based on the query (node or node set). For example, [38] introduces the notion of local conductance which penalizes the inclusion of nodes outside the query node set. The state of the art of this line of research [49] reports a community that optimizes the local conductance, reporting a community that substantially overlaps with the input seed nodes, moreover, the complexity is polynomial to the total degree of the query nodes as opposed to the entire node set of the graph (local algorithm). The local density proposed in this paper falls in this line of research, our local algorithm optimizes the local density, reporting a community that is both local and dense.

Random walk [2, 3, 9, 10, 50–52] based clustering ranks nodes by performing random walks (with restart) from the query nodes. One can produce one [3, 51] or possibly multiple [2, 9] communities around the given query nodes. The random walk also records partial [50, 52] or complete [10] visiting history to improve the result quality. The state-of-the-art method MRW [10] can optionally engage multiple walkers if more than one community shall be detected and engages a complete visiting history for better quality.

Motif-based approaches (see survey [21]) are drastically different from the above two categories in a sense that they rely heavily on indexing and perform no optimization/online search. The index focuses on predefined subgraphs on which the frequencies of motifs such as edges [7], triangles [1, 29], cliques [17] and their variations [40] satisfy certain constraints. The relation between the motif-based constraints and metrics such as density and conductance is loose (2-approximated density [7]), and the reported community can include a large number of nodes irrelevant to the query.

Table 2: The Description of the Datasets

Name	n	m	Density	Type
amazon [5]	334,863	925,872	2.76	Product network
notredame [6]	325,730	1,090,108	3.35	Hyperlink network
digg [6]	279,631	1,548,126	5.54	Social network
citeseer [6]	384,414	1,736,145	4.52	Citation network
livemocha [6]	104,103	2,193,083	21.07	Social network
flickr [5]	105,939	2,316,948	21.87	Image network
hyves [6]	1,402,674	2,777,419	1.98	Social network
youtube [5]	1,134,890	2,987,624	2.63	Social network
google [6]	875,714	4,322,051	4.94	Hyperlink network
trec [6]	1,601,788	6,679,248	4.17	Hyperlink network
flixster [6]	2,523,387	7,918,801	3.14	Social network
dblp [6]	1,653,767	8,159,739	4.93	Citation network
skitter [6]	1,696,416	11,095,299	6.54	Computer network
indian [6]	1,382,868	13,591,473	9.83	Hyperlink network
pokec [6]	1,632,804	22,301,964	13.66	Social network
usaroad [6]	23,947,347	28,854,312	1.20	Road network
livejournal [5]	3,997,962	34,681,189	8.67	Social network
orkut [5]	3,072,441	117,185,083	38.14	Social network
wikipedia [6]	13,593,033	334,591,525	24.61	Hyperlink network
friendster [6]	68,349,466	1,811,849,342	26.51	Social network
uk2007 [6]	105,153,952	3,301,876,564	31.40	Hyperlink network

6 EXPERIMENT

6.1 Compare ADS Algorithms: Local vs Global

Algorithms. This subsection compares three algorithms proposed in the paper. The input of the algorithms is a query $Q(G, A, R)$ where G is an undirected and unweighted graph while $R \subseteq V$ is a subset of nodes of G and $A \subseteq R$. All algorithms return the R-densest subgraph of G and compute max s-t flow with the push-relabel algorithm and were implemented in Julia.

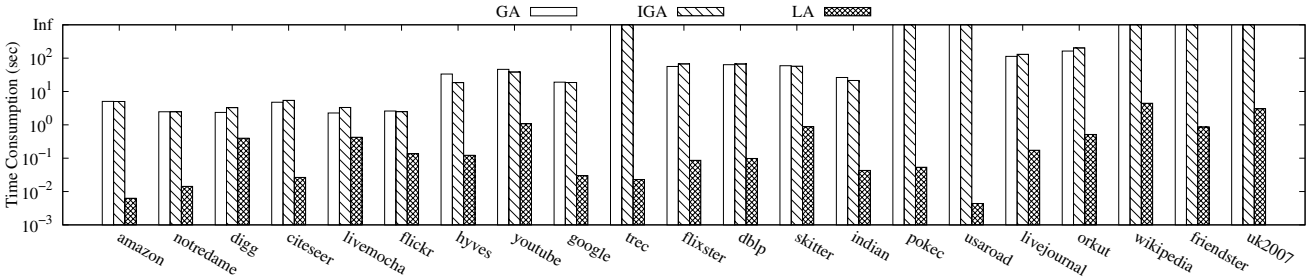


Figure 5: The Comparison of the Computation Time

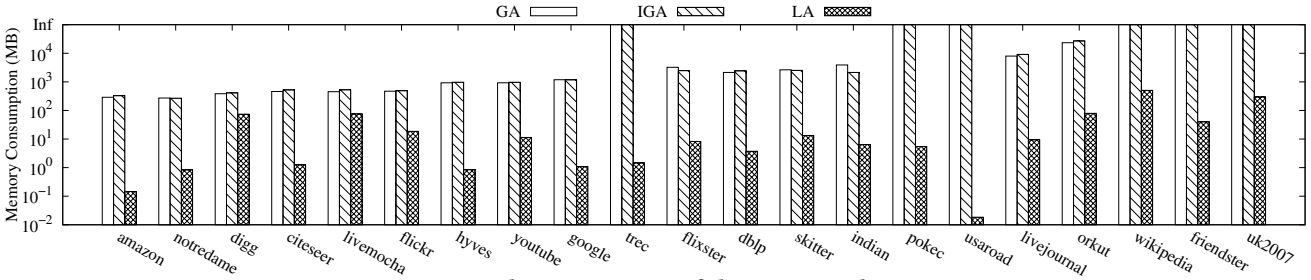


Figure 6: The Comparison of the Consumed Space

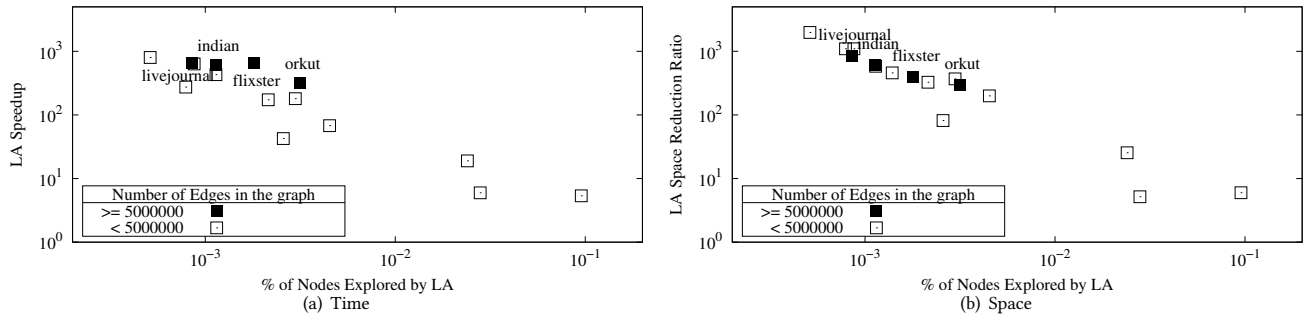


Figure 7: The Performance Gain of LA over GA on Different Data Graphs

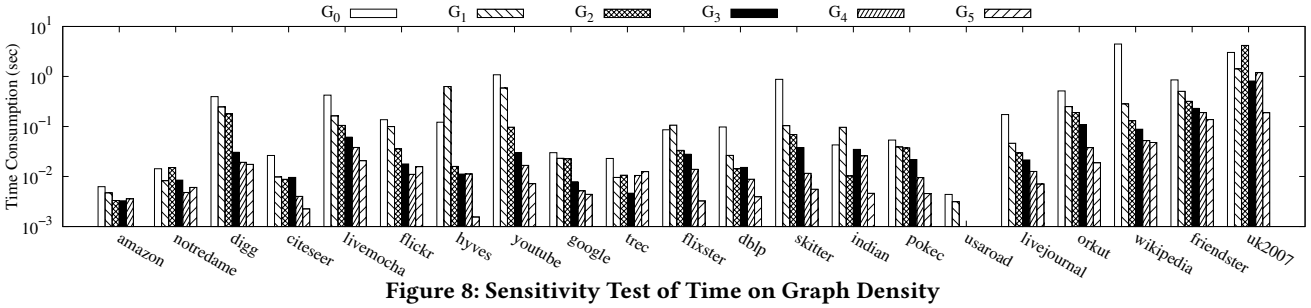


Figure 8: Sensitivity Test of Time on Graph Density

(1) Global Algorithm (GA): Algorithm 1 from Section 3.
(2) Improved Global Algorithm (IGA): merge all nodes in G with degree $\geq \text{vol}(R)$ to the target node t before applying GA. The correctness is based on Lemma 4.6.
(3) Local Algorithm (LA): Algorithm 2 in Section 4. It performs a series of max s-t flows on subgraphs of G grown around R.

Datasets. The experiments were carried out on 21 real graphs, all with more than 1 million edges. Table 2 describes these graphs. The largest graph, uk2007, contains over 100 million nodes and 3.3 billion edges. The data graphs were downloaded from Stanford Large Network Dataset Collection[5] [35] and KONECT Project[6] [32]. The graphs were preprocessed to be undirected, unweighted, selfloops-free and connected (we keep the largest connected component).

Query Generation. In real applications, the anchored node set A and the reference node set R should be provided by the user or the system (see use cases UC_1 and UC_2 in the introduction). To imitate the formation of A and R in generating a query, we choose a "user" v uniformly at random from V and then decide A and R.

[5] http://snap.stanford.edu/data/
[6] http://konect.cc/

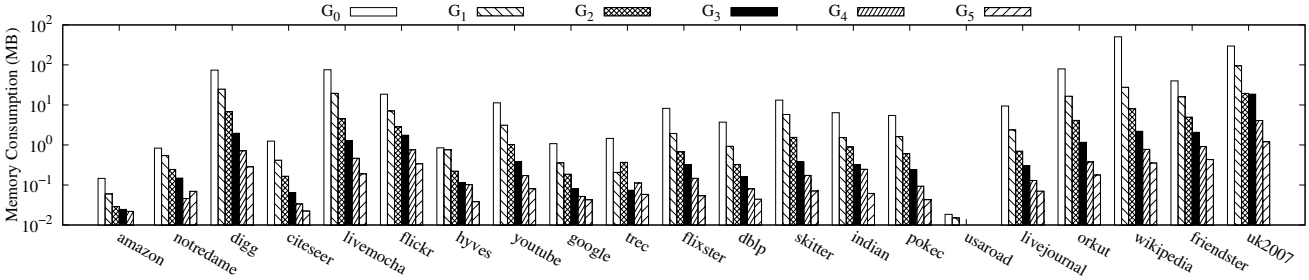


Figure 9: Sensitivity Test of Space on Graph Density

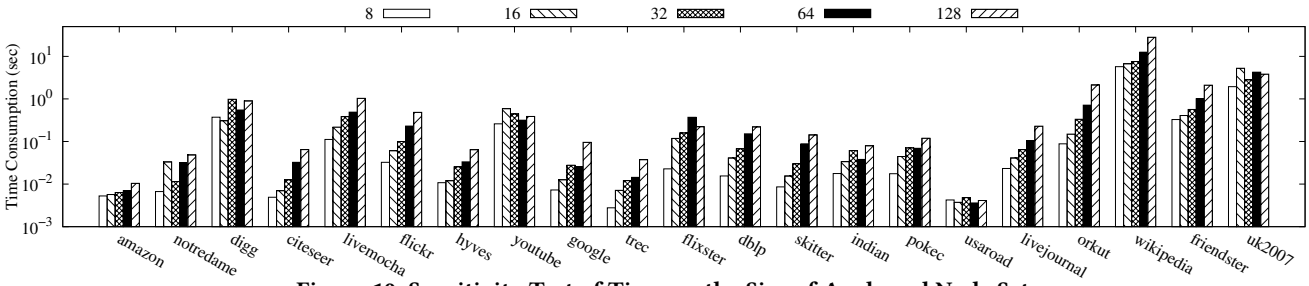


Figure 10: Sensitivity Test of Time on the Size of Anchored Node Set

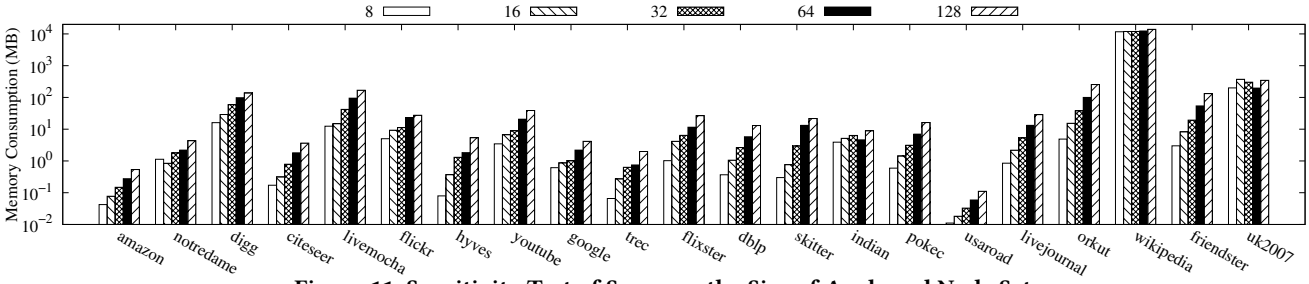


Figure 11: Sensitivity Test of Space on the Size of Anchored Node Set

A Select a fixed number of nodes from v's 1-hop and 2-hop neighbors uniformly at random to form A. Let $v \in A$. By default $|A| = 8$.

R From each node u in A, perform a fixed number (default 3) of random walks each of a fixed length (default 2). If the random walk visits a node u', $R \leftarrow R \cup \{u'\}$ if u' passes the *locality test on centrality*, that is, the degree of u' is no higher than $cap(A)$. $cap(A)$ is a function of the highest degree $d_{\max}(A) = \max_{v \in A}\{d_G(v)\}$ of A, i.e., $cap(A) = \mathrm{d_c} \cdot d_{\max}(A)$. The multiplier $\mathrm{d_c}$ is set so that when $d_{\max}(A)$ is very small, the random walk can be relaxed in taking the multihop neighbors of A. Specifically, when $d_{\max}(A)$ is small, e.g., $< |V|/e^{10}$, then we set $\mathrm{d_c} = 10$; otherwise, we set $\mathrm{d_c}$ based on $d_{\max}(A)$ and $|V|$, i.e., let $\mathrm{d_c} = \lceil \log(|V|/d_{\max}(A)) \rceil$.

The above query expansion process ensures that $A \subseteq R$. For each data graph, we generate a set of 100 queries. It turns out that the average size of R generated for all the real graphs is 42.

Environment. All experiments were conducted on a server with Intel Xeon Gold 6230 CPU 2.10GHz and 376 GB RAM running Ubuntu 5.8.0-38-Generic. Before testing on a data graph, we loaded the data graph into the main memory using a build-in function of Julia to support the processing of all the queries related to the graph. Then each algorithm was run over all the queries of the graph and reported the consumed time (allocated space resp.) for processing each query. The cutoff time was 1000 seconds for each query.

Exp-1: Performance on Different Data Graphs. We ran Algorithms GA, IGA, LA on the 21 real word graphs.

Figure 5 shows the running time and Figure 6 the memory usage. LA completed the computation on all the data graphs; GA and IGA failed on pokec, trec, usaroad and wikipedia by exceeding the cutoff time. GA and IGA failed on friendster and uk2007 with out-of-memory error. This means that for the large graph such as friendster and uk2007, the server can afford the memory for storing the graph but not the overhead of running max s-t flow on the graph since the flow algorithm is known to have a high constant factor.

Over all the data graphs that all the three algorithms completed, LA outperforms the other two by up to three orders of magnitude on both time consumption and memory usage. On average, LA reduces the running time of GA by a factor 283 and the space consumption by a factor of 425. GA and IGA has a similar performance, because IGA will only merge an average of 0.0046% of the graph nodes, leading to marginal performance gain of IGA over GA.

Unlike GA and IGA, the local algorithm LA does not have its time and memory costs increase with the graph size. Its costs are more related to the graph density. For example, pokec and usaroad have a similar number of edges and drastically different densities, 13.66 and 1.2, resp., the costs of LA on usaroad are significantly smaller than that on pokec. Exp-2 further explains this correlation.

Figure 7(a) and Figure 7(b) show the performance gain (time and space) of LA over GA. The x-axis marks the percentage of the nodes explored by LA, *i.e.*, (the # of nodes in G_α)/(the # of nodes in G). Each square represents the average performance gain (speedup in running time and space reduction ratio) over the queries of a data graph. The squares of large graphs, *i.e.*, with $\geq 5 \times 10^6$ nodes (flixster, indian, livejournal and orkut) are filled in black. The result shows that the performance gain of LA over GA is inversely proportional to the percentage of nodes explored by LA. Besides, the larger the graph is the higher the performance gain becomes. Moreover, without exception on large graphs (see the black squares), the speedups are all above 317 and the space reduction ratio above 296. This showcases the scalability of LA.

Exp-2: Sensitivity of LA on Graph Density. For each graph, we controlled the graph density in the following way. Denote by G_0 the original graph. Generate $G_1, \cdots, G_5$ where G_i is the largest connected component of the graph generated by including each edge in G_0 with probability 0.5^i. The query set including 1000 queries was generated under default parameters. If G_i is too small to include more than 128 nodes, we skip the graph. We ran LA for each query on each graph to record the time and memory.

Figure 8 and Figure 9 show the performance of LA in terms of time and memory usage. The results show that the time and space costs increase with the density. On average, when the density doubles, the time is increased by a factor of 2.037, and the space cost is increased by a factor of 2.13.

Exp-3: Sensitivity of LA on Reference Node Set Size. We generated one query set for each k, for $k = 8, 16, 32, 64, 128$, respectively, by repeating the following process 100 times. Firstly, we generated A with default procedure except that the size of A takes $\frac{k}{4}$. After that, we generated R by random walking (of length 2) from a random node in A until $|R|$ becomes k. Still, a *locality test* on node centrality applied. 5 query sets were generated for 5 different values of k. Figures 10-11 show that the time and space taken increase with $|R|$. On average, when $|R|$ doubles, the time cost is increased by a factor of 1.637 and the space is increased by a factor of 1.613.

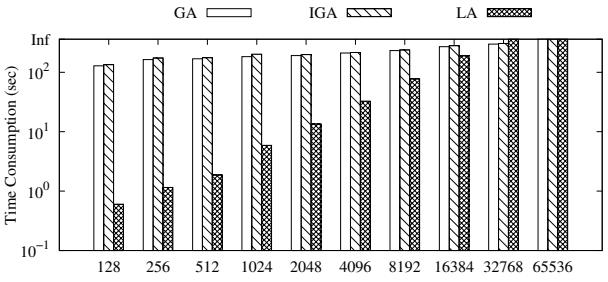


Figure 12: Query Time on Large Anchored Node Sets

Exp-4: Query Time on Larger Reference Node Sets. Varying the size of R from 128, we generated a sequence of reference node sets with exponentially increasing size, i.e., 256, 512, $\cdots$, using the same query expansion process on orkut and ran the three algorithms GA, IGA and LA. We use 6 minutes as the cut off time for query processing. Figure 12 shows the query time. Note that when $|R|$ becomes 65536, none of the algorithms can complete query processing before the cut off. Along with the doubling of $|R|$, the time cost of GA, IGA and LA increase by a factor of 1.113, 1.111 and 2.305, respectively, on average. LA is outperformed by GA and IGA when $|R| \geq 32768$. When $|R| > 10^4$, none of the proposed methods could efficiently process the query; however, in the use cases in Section 1 and Section 6.3, the R provided by the user or the system (based on a user's behavior) should reasonable stay within 10^3.

6.2 Compare to other Local Community Search

This section compares anchored densest subgraph search with existing local community detection problems in the literature in result quality and computation efficiency on all the graphs in Table 2.

Problems and Algorithms. We compare our proposed approach with the state-of-the-art local community detection algorithms (Section 5): one [49] optimizes the local conductance and the other [10] is based on random walk. All the algorithms were implemented in Julia in the environment described in Section 6.1.

(1) LA: Our proposed anchored densest subgraph search.
(2) FS: The flowseed algorithm [49] is the state-of-the-art metric optimization local community search algorithm. It optimizes the *local conductance* $\pi_R(S) = \frac{|E(S,\bar{S})|}{\text{vol}(R\cap S) - \epsilon\text{vol}(S\cap\bar{R}) - \sum_{u\in R, u\notin S} p_u d(u)}$ for a given reference node set R. Unlike $\rho_R(S)$, the term $\sum_{u\in R, u\in\bar{S}} p_u d(u)$ penalizes the nodes in R that are not included in S which $\rho_R(S)$ does not penalize. Therefore, for a fair comparison, we set parameter $p_u = 0$, for $\forall u \in V$, and set $\epsilon = 1$ to be parameter free. We used the code[7] provided by the authors of [49] in Julia.
(3) MRW: The MRW algorithm [10] is the state-of-the-art random walk based local community detection algorithm. As our objective is to find a single community, we used the single random walker version of MRW. We translated the code to Julia using their default parameters $\alpha = 0.1$, $\beta = 0.6$ and $K = 5$. The random walk restarts from a node sampled uniformly at random from R. We report the nodes ranked top-15 by MRW to form the resulting subgraph. The size is set to 15 to be comparable to the results of the other two approaches (Figure 13(f)).

Query Generation. For each data graph, we generate a set of 100 queries. As MRW does not support anchored node set, we set the anchored node set $A = \emptyset$. Similar to the query expansion described in Section 6.1, we select a "user" x uniformly at random from V and then generate the reference node set R by applying 15 random walks from x up to 4 hops each.

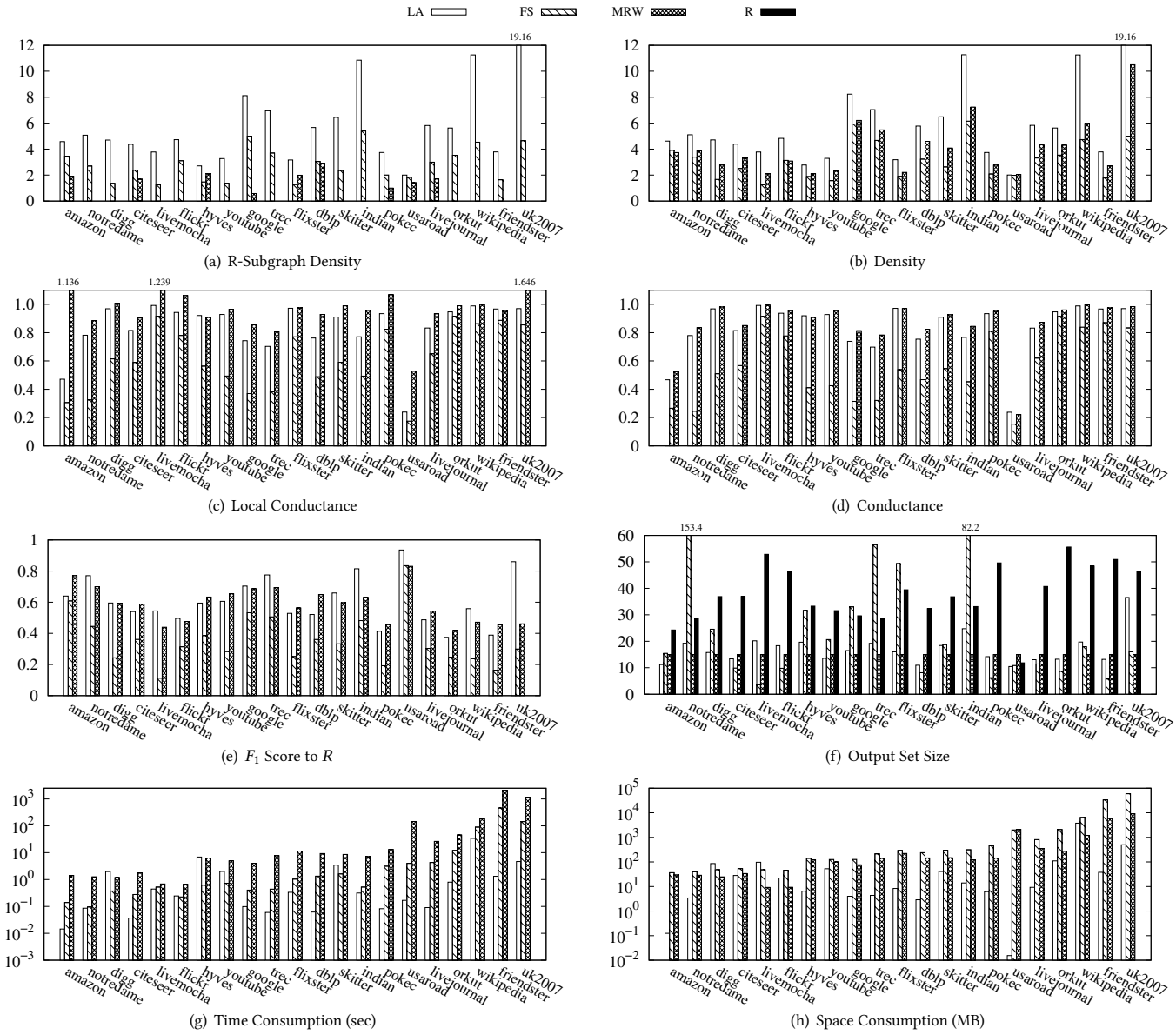
Evaluation. We measure each reported subgraph S with

- Two local community metrics: R-**subgraph density** from our paper and **local conductance** proposed by [49],
- Two baseline community metrics: **density** and **conductance**,
- The size $|S|$ of the subgraph,
- Measure of locality: use R as the ground truth to compute the F_1-score $(2|S \cap R|)/(|S| + |R|)$ of S.

Figure 13 shows the results of LA, FS and MRW on the 21 graphs.

Density. Figures 13(a)–13(b) show that the average R-subgraph density of LA is higher than that of FS by 113%. MRW is not visible on the majority of the data graphs due to its negative R-subgraph densities. The average density of the results of LA is higher than that of FS and MRW by 92% and 48%, respectively.

[7] https://github.com/nveldt/FlowSeed



(a) R-Subgraph Density

(b) Density

(c) Local Conductance

(d) Conductance

(e) F_1 Score to R

(f) Output Set Size

(g) Time Consumption (sec)

(h) Space Consumption (MB)

Figure 13: Comparison with Other Local Community Search

Conductance. Figures 13(c)–13(d) show that FS has the lowest (local) conductance among the three, which is expected since FS is an optimization algorithm of the local conductance. LA outperforms MRW: the local conductance of LA is lower than MRW by 15.3% and the conductance of LA is lower than MRW by 3.3%.

Locality. Figure 13(e) shows that LA has the highest locality among the three algorithms: the F_1-score of LA is 71% higher than FS and 4% higher than MRW on average.

Output Size. Figure 13(f) shows that the average output size of LA is 17.0, the output size of MRW is fixed at 15. Apart from the two graphs notredame and indian where FS has significantly large output size, the average output size of FS is 18.8.

Computation Time. Figure 13(g) shows that LA outperforms the baselines: LA is 12.5× faster than FS and is 66.7× faster than MRW.

Memory Consumption. Figure 13(h) shows that LA uses up to one order of magnitude less memory than the baseline: the memory consumption of LA is $\frac{1}{20}$ that of FS and is $\frac{1}{4.3}$ that of MRW.

6.3 Case Study

This section compares the effectiveness of LA, FS and MRW in two real-world applications described in Section 1, i.e., UC_1 event organization and UC_2 product recommendation.

UC_1. Consider collaboration network dblp (Table 2) where each node represents an author and each undirected edge denotes that two authors have co-authored at least one article. The case study adopted dblp for its availability of people's names; similar applications can be deployed on social or professional networks.

Let x be a person. Let R be the set of people whom x may have known well. x can know a person by, in addition to co-authoring

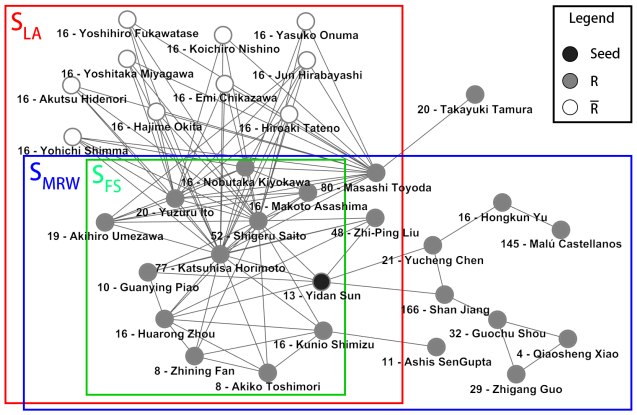


Figure 14: Local Communities Near *Sun Yidan*

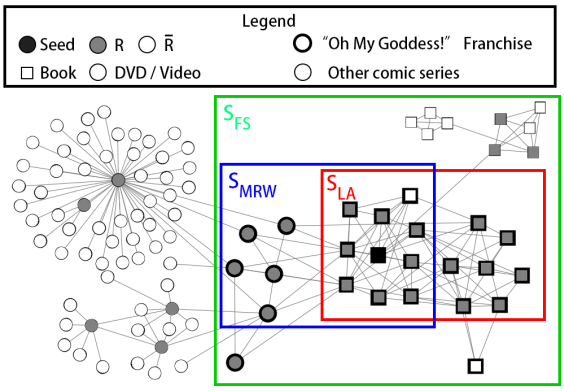


Figure 15: Local Communities Near *Oh My Goddess!*

an article, attending the same conference, joining the same lab, etc.. The reference node set R should be provided by the user x; in our case study, we generated R from x in the same way as Section 6.2. With R, we ran the three local community detection algorithms, LA, FS and MRW (following Section 6.2, MRW reports top-15 ranked nodes). The detected community can be used for organizing a social event which may help x expand his/her collaboration network.

Figure 14 shows the results when x takes an ordinary node, an author named *Yidan Sun* (black color), who has 13 co-authors on dblp. The reference node set R generated is colored in grey and the communities returned by LA, FS and MRW are in rectangles with labels S_{LA}, S_{FS}, and S_{MRW}, respectively. Each node is labeled with its degree while edges to non-displayed nodes are not manifested.

It can be observed that S_{LA} includes 10 nodes from $\overline{R}$ which i) have strong relations with the nodes in $R \cap S_{\text{LA}}$ and ii) have not been identified by S_{FS} and S_{MRW}. The node degree within the subgraph S_{LA} is high, indicating potential connections to be established to x, thus S_{LA} forms an ideal list of attendees if x would like to host a social event. S_{FS} reports the smallest subgraph among the three. All the nodes in S_{MRW} are from R, including the nodes who do not have a strong connection to the other nodes in S_{MRW}, e.g., Hongkun Yu, and nodes who have significantly stronger connection (144/145) to $V \setminus S_{\text{MRW}}$, e.g., Malú Castellanos, these people might find themselves hard to fit in a social event with people in S_{MRW}.

UC$_2$. Amazon co-purchase network [34], or amazon for short, is a network that is based on *Customers Who Bought This Item Also Bought* feature of the Amazon website. Each node represents a product, and each undirected edge represents two products that have been bought together. The data set was downloaded from The KONECT Project (http://konect.cc/networks/com-amazon/). The data was collected in 2006 on products including books, DVDs, musics and videos. This study considers the largest connected component of the data graph of 334, 863 nodes and 925, 872 edges.

Let x be an item that a user put into the shopping cart and R the user's visiting history which should be provided by the system. Here we generated R following the same method of Section 6.2, ran the three algorithms with R. The results can be used for product recommendation or other marketing strategies.

Figure 15 shows the output of the three algorithms with rectangles labeled S_{LA}, S_{FS} and S_{MRW}, respectively, when x is *Oh My Goddess!: Final Exam (Book)*, one of a Japanese comic book series *Oh My Goddess!*. By the time the data was collected, the comic series has more than 20 volumes available on Amazon, there are also DVDs available on Amazon under the same franchise. We use square nodes to denote books, round nodes DVDs and videos. Nodes with thick borders are products of *Oh My Goddess!* franchise, and nodes with thin border belong to other comic series. x is in black filling, nodes in R grey filling and other nodes no filling.

It can be observed that LA produces the densest community among the three, FS produces the largest subgraph with the smallest number of cut edges (between S_{FS} and $\overline{S_{\text{FS}}}$). This can be explained by the different objectives optimized by the two algorithms. Content-wise, S_{LA} contains only the comic books of *Oh My Goddess!* series. S_{FS}, in addition to items in S_{LA}, contains DVDs and Videos of *Oh My Goddess!* series and some other comic book titles. S_{MRW} is a hybrid of the other two: of *Oh My Goddess!*, it misses some comic books but includes some DVDs. The LA can be used for product recommendation according to the result since people who have bought a comic book are likely to have interest in other volumes of the same comic book series.

Remarks. The case studies show that for local community detection, our algorithm LA produces a community that is biased towards a reference node set and is more coherent than that of FS and MRW, attributing to the optimized density of the resulting subgraph.

7 CONCLUSION

Given a graph G and a reference node set R, this paper proposes anchored densest subgraph search (ADS) which penalizes non-R nodes proportional to their centralities. The paper also proposes, for ADS, a local (search) algorithm whose complexity is related to R as opposed to the entire graph G. Extensive experiments have verified the efficiency and effectiveness of the proposed algorithm. Use cases are provided to apply the techniques to real-life scenarios. Approximation algorithms for efficient ADS will be investigated as future work.

ACKNOWLEDGMENTS

Miao Qiao is supported by Marsden Fund UOA1732, and MBIE Catalyst: Strategic Fund NZ-Singapore Data Science Research Programme UOAX2001. Lijun Chang is supported by the Australian Research Council Funding of FT180100256 and DP220103731.

REFERENCES

[1] Esra Akbas and Peixiang Zhao. 2017. Truss-based community search: a truss-equivalence based indexing approach. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1298–1309.

[2] Morteza Alamgir and Ulrike Von Luxburg. 2010. Multi-agent random walks for local clustering on graphs. In *2010 IEEE International Conference on Data Mining*. IEEE, 18–27.

[3] Reid Andersen, Fan Chung, and Kevin Lang. 2006. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. IEEE, 475–486.

[4] Mehdi Azaouzi, Delel Rhouma, and Lotfi Ben Romdhane. 2019. Community detection in large-scale social networks: state-of-the-art and future directions. *Social Network Analysis and Mining* 9, 1 (2019), 1–32.

[5] B. Bahmani, R. Kumar, and S. Vassilvitskii. 2012. Densest Subgraph in Streaming and MapReduce. *Proceedings of the VLDB Endowment (PVLDB)* 5, 5 (2012), 454–465.

[6] O. D. Balalau, F. Bonchi, T.-H. H. Chan, F. Gullo, and M. Sozio. 2015. Finding Subgraphs with Maximum Total Density and Limited Overlap. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, WSDM 2015, Shanghai, China, February 2-6, 2015*. ACM, 379–388.

[7] Nicola Barbieri, Francesco Bonchi, Edoardo Galimberti, and Francesco Gullo. 2015. Efficient and effective community search. *Data mining and knowledge discovery* 29, 5 (2015), 1406–1433.

[8] S. Bhattacharya, M. Henzinger, D. Nanongkai, and C. E. Tsourakakis. 2015. Space- and Time-Efficient Algorithm for Maintaining Dense Subgraphs on One-Pass Dynamic Streams. In *Proc. of STOC'15*. 173–182.

[9] Yuchen Bian, Jingchao Ni, Wei Cheng, and Xiang Zhang. 2017. Many heads are better than one: Local community detection by the multi-walker chain. In *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 21–30.

[10] Yuchen Bian, Yaowei Yan, Wei Cheng, Wei Wang, Dongsheng Luo, and Xiang Zhang. 2018. On multi-query local community detection. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 9–18.

[11] Tanmoy Chakraborty, Ayushi Dalmia, Animesh Mukherjee, and Niloy Ganguly. 2017. Metrics for community analysis: A survey. *ACM Computing Surveys (CSUR)* 50, 4 (2017), 1–37.

[12] Lijun Chang and Miao Qiao. 2020. Deconstruct Densest Subgraphs. In *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, Yennun Huang, Irwin King, Tie-Yan Liu, and Maarten van Steen (Eds.). ACM / IW3C2, 2747–2753.

[13] Lijun Chang and Lu Qin. 2018. *Cohesive Subgraph Computation over Large Sparse Graphs*. Springer Series in the Data Sciences.

[14] M. Charikar. 2000. Greedy approximation algorithms for finding dense components in a graph. In *Approximation Algorithms for Combinatorial Optimization, Third International Workshop*. 84–95.

[15] Jiyang Chen, Osmar Zaïane, and Randy Goebel. 2009. Local community identification in social networks. In *2009 International Conference on Advances in Social Network Analysis and Mining*. IEEE, 237–242.

[16] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. 2001. *Introduction to Algorithms*. McGraw-Hill Higher Education.

[17] Wanyun Cui, Yanghua Xiao, Haixun Wang, Yiqi Lu, and Wei Wang. 2013. Online search of overlapping communities. In *Proceedings of the 2013 ACM SIGMOD international conference on Management of data*. 277–288.

[18] M. Danisch, T.-H. H. Chan, and M. Sozio. 2017. Large Scale Density-friendly Graph Decomposition via Convex Programming. In *Proc. of WWW'17*. 233–242.

[19] Y. Dourisboure, F. Geraci, and M. Pellegrini. 2007. Extraction and classification of dense communities in the web. In *Proceedings of the 16th International Conference on World Wide Web, WWW*. ACM, 461–470.

[20] Alessandro Epasto, Silvio Lattanzi, and Mauro Sozio. 2015. Efficient Densest Subgraph Computation in Evolving Graphs. In *Proc. of WWW'15*. 300–310.

[21] Yixiang Fang, Xin Huang, Lu Qin, Ying Zhang, Wenjie Zhang, Reynold Cheng, and Xuemin Lin. 2020. A survey of community search over big graphs. *The VLDB Journal* 29, 1 (2020), 353–392.

[22] Yixiang Fang, Kaiqiang Yu, Reynold Cheng, Laks V. S. Lakshmanan, and Xuemin Lin. 2019. Efficient Algorithms for Densest Subgraph Discovery. *PVLDB* 12, 11 (2019), 1719–1732.

[23] K. Fountoulakis, M. Liu, D. F. Gleich, and M. W. Mahoney. 2020. Flow-based Algorithms for Improving Clusters: A Unifying Framework, Software, and Performance. *CoRR* abs/2004.09608 (2020).

[24] Esther Galbrun, Aristides Gionis, and Nikolaj Tatti. 2016. Top-k overlapping densest subgraphs. *Data Min. Knowl. Discov.* 30, 5 (2016), 1134–1165.

[25] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan. 1989. A Fast Parametric Maximum Flow Algorithm and Applications. *SIAM Journal of Computing* 18, 1 (1989), 30–55.

[26] D. Gibson, R. Kumar, and A. Tomkins. 2005. Discovering Large Dense Subgraphs in Massive Graphs. In *Proceedings of the 31st International Conference on Very Large Data Bases*. ACM, 721–732.

[27] Aristides Gionis and Charalampos E. Tsourakakis. 2015. Dense Subgraph Discovery: KDD 2015 tutorial. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, Longbing Cao, Chengqi Zhang, Thorsten Joachims, Geoffrey I. Webb, Dragos D. Margineantu, and Graham Williams (Eds.). ACM, 2313–2314.

[28] A. V. Goldberg. 1984. *Finding a Maximum Density Subgraph*. Technical Report UCB/CSD-84-171. EECS Department, University of California, Berkeley.

[29] Xin Huang, Hong Cheng, Lu Qin, Wentao Tian, and Jeffrey Xu Yu. 2014. Querying k-truss community in large and dynamic graphs. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 1311–1322.

[30] Yasushi Kawase, Yuko Kuroki, and Atsushi Miyauchi. 2019. Graph Mining Meets Crowdsourcing: Extracting Experts for Answer Aggregation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, Sarit Kraus (Ed.). ijcai.org, 1272–1279.

[31] Samir Khuller and Barna Saha. 2009. On Finding Dense Subgraphs. In *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, Vol. 5555. 597–608.

[32] Jérôme Kunegis. 2013. KONECT – The Koblenz Network Collection. In *Proc. Int. Conf. on World Wide Web Companion*. 1343–1350. http://dl.acm.org/citation.cfm?id=2488173

[33] V. E. Lee, N. Ruan, R. Jin, and C. C. Aggarwal. 2010. A Survey of Algorithms for Dense Subgraph Discovery. In *Managing and Mining Graph Data*. 303–336.

[34] Jure Leskovec, Lada A Adamic, and Bernardo A Huberman. 2007. The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)* 1, 1 (2007), 5–es.

[35] J. Leskovec and A. Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[36] Chenhao Ma, Yixiang Fang, Reynold Cheng, Laks V. S. Lakshmanan, Wenjie Zhang, and Xuemin Lin. 2020. Efficient Algorithms for Densest Subgraph Discovery on Large Directed Graphs. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*. 1051–1066.

[37] Stefanie Muff, Francesco Rao, and Amedeo Caflisch. 2005. Local modularity measure for network clusterizations. *Physical Review E* 72, 5 (2005), 056107.

[38] L. Orecchia and Z. A. Zhu. 2014. Flow-Based Algorithms for Local Graph Clustering. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, C. Chekuri (Ed.). SIAM, 1267–1286.

[39] Lu Qin, Rong-Hua Li, Lijun Chang, and Chengqi Zhang. 2015. Locally Densest Subgraph Discovery. In *Proc. of KDD'15*. 965–974.

[40] Jing Shan, Derong Shen, Tiezheng Nie, Yue Kou, and Ge Yu. 2016. Searching overlapping communities for group query. *World Wide Web* 19, 6 (2016), 1179–1202.

[41] Bintao Sun, Maximilien Danisch, T.-H. Hubert Chan, and Mauro Sozio. 2020. KClist++: A Simple Algorithm for Finding k-Clique Densest Subgraphs in Large Graphs. *Proc. VLDB Endow.* 13, 10 (2020), 1628–1640.

[42] Nikolaj Tatti and Aristides Gionis. 2013. Discovering Nested Communities. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 8189)*, Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Železný (Eds.). Springer, 32–47.

[43] N. Tatti and A. Gionis. 2015. Density-friendly Graph Decomposition. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*. ACM, 1089–1099.

[44] N. Tatti and A. Gionis. 2015. Density-friendly Graph Decomposition. In *Proc. of WWW'15*. 1089–1099.

[45] Charalampos E. Tsourakakis. 2015. The K-clique Densest Subgraph Problem. In *Proc. of WWW'15*. 1122–1132.

[46] Elena Valari, Maria Kontaki, and Apostolos N. Papadopoulos. 2012. Discovery of Top-k Dense Subgraphs in Dynamic Graph Collections. In *Scientific and Statistical Database Management - 24th International Conference, SSDBM 2012, Chania, Crete, Greece, June 25-27, 2012. Proceedings (Lecture Notes in Computer Science, Vol. 7338)*, Anastasia Ailamaki and Shawn Bowers (Eds.). Springer, 213–230.

[47] E. Valari, M. Kontaki, and A. N. Papadopoulos. 2012. Discovery of Top-k Dense Subgraphs in Dynamic Graph Collections. In *Proc. of SSDBM'12*. 213–230.

[48] Twan Van Laarhoven and Elena Marchiori. 2016. Local network community detection with continuous optimization of conductance and weighted kernel k-means. *The Journal of Machine Learning Research* 17, 1 (2016), 5148–5175.

[49] Nate Veldt, Christine Klymko, and David F Gleich. 2019. Flow-based local graph clustering with better seed set inclusion. In *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM, 378–386.

[50] Yubao Wu, Yuchen Bian, and Xiang Zhang. 2016. Remember where you came from: on the second-order random walk based proximity measures. *Proceedings of the VLDB Endowment* 10, 1 (2016), 13–24.

[51] Yubao Wu, Ruoming Jin, Jing Li, and Xiang Zhang. 2015. Robust local community detection: on free rider effect and its elimination. *Proceedings of the VLDB Endowment* 8, 7 (2015), 798–809.

[52] Yubao Wu, Xiang Zhang, Yuchen Bian, Zhipeng Cai, Xiang Lian, Xueting Liao, and Fengpan Zhao. 2018. Second-order random walk-based proximity measures in graph analysis: formulations and algorithms. *The VLDB Journal* 27, 1 (2018), 127–152.