Finding Top-r Influential Communities under Aggregation Functions

You Peng*, Song Bian*, Rui Li, Sibo Wang, Jeffrey Xu Yu

The Chinese University of Hong Kong
{ypeng,sbian,lirui,swang,yu}@se.cuhk.edu.hk

***Abstract*—Community search is a problem that seeks cohesive and connected subgraphs in a graph that satisfy certain topology constraints, e.g., degree constraints. The majority of existing works focus exclusively on the topology and ignore the nodes' influence in the communities. To tackle this deficiency, influential community search is further proposed to include the node's influence. Each node has a weight, namely influence value, in the influential community search problem to represent its network influence. The influence value of a community is produced by an aggregated function, e.g., max, min, avg, and sum, over the influence values of the nodes in the same community. The objective of the influential community search problem is to locate the top-r communities with the highest influence values while satisfying the topology constraints. Existing studies on influential community search have several limitations: *(i)* they focus exclusively on simple aggregation functions such as min, which may fall short of certain requirements in many real-world scenarios, and *(ii)* they impose no limitation on the size of the community, whereas most real-world scenarios do. This motivates us to conduct a new study to fill this gap.**

We consider the problem of identifying the top-r influential communities with/without size constraints while using more complicated aggregation functions such as sum or avg. We give a theoretical analysis demonstrating the hardness of the problems and propose efficient and effective heuristic solutions for our top-r influential community search problems. Extensive experiments on real large graphs demonstrate that our proposed solution is significantly more efficient than baseline solutions.

I. Introduction

In reality, graph data becomes increasingly complicated and diverse. The vertex of the graph is filled with relevant information [1]–[10]. The information could be obtained from either raw data, e.g., H-index and income, or from the topological structure of the graph, e.g., PageRank, Closeness, Degree, and Betweenness. For instance, Twitter could be abstracted into a graph, with each vertex representing a user, and each edge representing whether two individuals follow each other. Then, the information of each vertex could be represented by their influence values. Meanwhile, numerous networks have a community structure. The community structure has a wide range of applications in a variety of disciplines, including social network mining [11], [12], biology analysis [13], and financial markets [14], [15]. Thus, extracting community structure from a graph is a fundamental problem in graph mining.

A substantial body of previous works have concentrated exclusively on discovering cohesive subgraphs from a large graph, ignoring the attribute of each vertex. Considering this, some works [16], [17] investigate a new community model based on the concept of k-core [18], [19], which is utilized to locate top-r k-influential communities over massive graphs. Due to the existence of additional cohesive requirements, the new model is extended to include additional cohesiveness metrics, e.g., k-truss [20]. As mentioned in [16], [17], the existing influential communities are mainly based on the k-core model.

Although existing models and approaches are practical and effective, the influence value of community is determined by the minimum value of vertices in their models. This assumption limits their applications. Therefore, the existing model is not always capable of satisfying the users' requirements. Thus, we aim to investigate the top-r k-influential community, whose influence value is determined by real-world applications. Based on this intuition, we study a generic influential community model in this paper. The influential community should be constrained by the following criteria: (1) it is a connected subgraph; (2) each vertex of the subgraph has at least k neighbors; and (3) there does not exist a supergraph, such that the influence value of the supergraph is the same as that of it. Additionally, the supergraph satisfies (1) and (2). The influence value of the community, on the other hand, should be determined by various aggregation functions, e.g., avg, sum, over the entire community, rather than by the vertex's minimum influence value. Additionally, the community search results should be non-overlapping.

Applications. Some real-life scenarios are listed as follows to demonstrate our motivations.

(1) Engagement. It is common for team members' engagement [21]–[24] to be determined by the number of friends in the same group. Also, the ability of each member is different. When the team encountered a financial crisis, it was forced to lay off several members. The leader wishes to reduce the size of the squad while maintaining its strength. Then, we could abstract the relationship in the team as a graph and assign each node an influence value as their ability. By identifying the top-r k-influential community, we could determine who should be laid off.

In this application, max could retain the most critical members. By using $weight\ density$, a highly connected and influential community could be reserved. $Balanced\ density$ is a variant of $density$, which requires the lay-off members are also highly connected.

(2) Group Recommendation. In a social network, a user may choose to search groups with similar interests in social networks [25]–[27]. For instance, a user could search for

*Equal contribution

TABLE I: Aggregation Functions under k-core Model

Aggregation functions	Formulas $f(H)$	Hardness
Minimum	$\min_{v\in H} w(v)$	P
Maximum	$\max_{v\in H} w(v)$	P
Sum	$w(H)=\sum_{v\in H} w(v)$	P
Sum-surplus	$w(H)+\alpha\|H\|$	P
Average	$w(H)/\|H\|$	NP-hard
Weight Density	$w(H)-\beta\|H\|$	NP-hard
Balanced Density	$\frac{w(H)}{w(H)-w(V\setminus H)}$	NP-hard

keywords on Facebook[1] or Twitter[2] to discover several communities with similar interests. We may assign a similar value to each user's influence in such a social network. The user then seeks out a community with the maximum influence value. The influence value of the community is determined by the average of its members.

(3) Influential Research Groups Identification. Mining a research community has been studied for more than two decades [16], [17], [28]–[33]. To locate influential research groups in a research network, e.g., DBLP[3], we could use the influence value of each vertex as the H-index and extract a community with a maximum influence value. Nevertheless, it is worth noting that a significant number of freshly graduated students have joined the group as new professors recently.

In this application, min and avg could be used to discover a group of highly cited researchers. Nevertheless, they are suitable for different citation metrics, e.g., i-10 index, G index. It could be seen from the case study in Section VI.C that G-$index$ is suitable for avg, while i-10 index is appropriate for min. As for sum, it could discover high-quality research community with more diversity.

Inspired by the aforementioned scenarios, it is necessary to investigate the top-r k-influential community search under various aggregation functions with or without size constraint, which could solve many real-life issues.

In local community detection, a goodness metric is usually used to measure whether a subgraph forms a community. The existing goodness metrics for local community detection can be categorized into three classes. The first class optimizes the internal denseness of a subgraph, i.e., the set of nodes in a community should be densely connected with each other. Such metrics include the classic density definition [34], edge-surplus [35], and minimum degree [36]. The second class optimizes both the internal denseness and the external sparseness. That is, the set of nodes in the community are not only densely connected with each other, but also sparsely connected with the nodes that are not in the community. Such metrics include subgraph modularity [37], density-isolation [38], and external conductance [39]. The local modularity measures the sharpness of the community boundary and belongs to the third class [40]. Using this metric, the set of nodes in the boundary of the community are highly connected to the nodes in the community but sparsely connected to the nodes outside the community.

[1]https://www.facebook.com/

[2]https://twitter.com/

[3]https://dblp.uni-trier.de/

Challenges and Contributions. The purpose of this paper is to investigate the problem of determining the top-r k-influential community over a massive graph using various aggregation functions, e.g., avg, sum. Table XIII lists a collection of commonly used aggregation functions[4]. We would primarily discuss the impact of aggregation functions on the top-r k-influential community search. However, in this study, we disregard the procedure of computing the weight of each vertex.

We have demonstrated that individuals occasionally like to select some influential communities with no overlaps or to identify several influential communities with size constraints. The top-r non-overlapping k-influential community search problem is investigated with or without size constraint.

By examining the top-r k-influential community search problem under various aggregation functions, we claim that the problem could be solved in polynomial time under some different aggregation functions, e.g., min, max. We develop a global search algorithm. Then, an improved algorithm is proposed for the problem if the aggregation function is sum. However, problems under some aggregation functions, e.g., avg, are NP-hard. Unless P = NP, they cannot be addressed in polynomial time. Thus, there are no solutions to these problems that are approximated by the constant-factor.

When the aggregation function is avg or sum, problems with size constrained are NP-hard. In light of this, we propose several efficient heuristic algorithms for the NP-hard problems based on local search. The main contributions of our paper are summarized as follows:

- *Various Aggregation Functions.* We extend the original influential community model to various aggregation functions. We analyze the hardness of the problem under different aggregation functions and propose efficient approaches for the influential community search problem.
- *Size-Constrained Influential Community.* We advocate a cohesive subgraph model: size-constrained influential community. We analyze the hardness of the new cohesive subgraph model, and propose some efficient heuristic algorithms. Additionally, we extend our approach to the top-r non-overlapping k-influential community search problems.
- *Efficiency and Effectiveness.* Extensive experiments on real networks demonstrate the efficiency of our techniques. In addition, a case study on a real dataset demonstrates the effectiveness of our model and algorithms.

Roadmap. The rest of the paper is organized as follows. Section II formally defines the problem. Section III provides hardness analysis of the top-r non-overlapping k-influential community search problems with or without size constraint. Solutions to top-r size-unconstrained (constrained) k-influential community problem are proposed in Section IV and V, respectively. followed by the empirical study in Section VI. Section VII surveys important related work. Section IX concludes the paper.

[4]The NP-hardness of Weight Density and Balanced Density is given in Appendix of our full version https://bit.ly/3Fa6YdW .

TABLE II: Notation Table

Notations	Description
$G(V, E, w)$	a weighted and undirected graph, where V is the set of vertices, E is the set of edges, and w is a weighted function
$G[H]$	the subgraph induced by H
n (m)	the number of nodes (edges) in G
k	the degree constraint for subgraph
s	the size constraint for subgraph
f	the aggregation function
$g(\cdot)$	the objective function of problem
$N(u, G)$	the set of neighbors of vertex u in G
$N(u, H)$	the set of neighbors of vertex u in $G[H]$
$d(u, G)$	the degree of vertex u in G
$d(u, H)$	the degree of vertex u in $G[H]$
$\delta(G)/\delta(H)$	the minimum degree of $G/G[H]$
$f(G)/f(H)$	the influence value of $G/G[H]$

II. PRELIMINARIES

In this section, we will begin by providing some basic background. Following that, we define the problems that will be discussed in this paper.

A. Problem Definitions

Let $G(V, E, w)$ be an undirected and weighted graph, where V denotes a set of vertices, $E \subseteq V \times V$ indicates a set of edges, and w is a weighted function that assigns each vertex $u \in V$ with a non-negative weight value. Throughout this work, we refer to $w(u, G)$ as the weight of vertex u in G. The weight assigned to each vertex could reflect the centrality of each vertex such as Pagerank, Betweenness, Closeness, or other attributes [16]. Moreover, $N(u, G) = \{v \in V | (u, v) \in E\}$ denotes the set of neighbors of vertices $u \in V$ in G. The degree of u is $d(u, G) = |N(u, G)|$. When the context is clear, we omit graph G in the notation. Table II lists the notations used in this paper.

This paper is mostly concerned with the k-core model. Let H be a subset vertices of V, which implies that $H \subseteq V$. The induced subgraph, denoted by $G[H] = (V_H, E_H, w)$, is a k-core if it conforms to the following definition:

Definition 1 (k-core)**.** *Given a graph $G = (V, E, w)$, a subgraph $G[H] = (V_H, E_H, w)$ is a k-core of G, if $G[H]$ satisfies the following constraints:*

1) ***Cohesive:*** *For any $u \in V_H$, $d(u) \geq k$.*
2) ***Maximal:*** *H is maximal, i.e., for any vertex set $H' \supset H$, $G[H']$ is not a k-core.*

In this paper, we aim to identify the influential communities in large networks. The influential community is a cohesive subgraph whose cohesiveness is based on k-core. The influential community has an influence value. Before introducing the concept of influential community, we refer to previous works [16], [17] that define the influence value of an induced subgraph. The definition is given below:

Definition 2. *($f(G[H])$). Let $G[H]$ be a subgraph of G, and f denotes an aggregation function. The influence value of subgraph $G[H]$ is denoted by $f(G[H])$, or simply $f(H)$ when the context is clear.*

Nevertheless, previous works [16], [17] concentrate exclusively on the influential community, where the influence value of a community is based on the minimum weight of the vertices in it. On the contrary, a user is more likely to find an influential community whose influence value is determined by various aggregation functions to solve the issues mentioned above. As a result, in contrast to previous research, we provide a general definition of influential community here.

Definition 3 (k-Influential Community)**.** *Given an undirected and weighted graph $G = (V, E, w)$, a vertex set $H \subseteq V$ and an aggregation function f, the induced subgraph $G[H] = (V_H, E_H, w)$ is a k-influential community if*

1) ***Cohesive:*** *For any $u \in V_H$, $d(u, H) \geq k$.*
2) ***Connected:*** *$G[H]$ is a connected subgraph.*
3) ***Maximal:*** *There is no other vertex set $H' \supset H$, such that induced subgraph $G[H'] = (V_{H'}, E_{H'}, w)$ satisfies 1) and 2), and $f(H') = f(H)$.*

Additionally, in certain real-life scenarios, we require a community with a limited size. Thus, we define the size-constrained influential community below.

Definition 4 (Size-Constrained k-Influential Community)**.** *Given a weighted and undirected graph $G = (V, E, w)$, the degree constraint k and the size constraint s. A size-constrained k-influential community $G[H] = (V_H, E_H, w)$ is a k-influential community with $|V_H| \leq s$.*

We principally focus on the following two influential community search problems in this paper:

Problem 1 (Top-r size-constrained k-Influential Community)**.** *Given a weighted and undirected graph $G = (V, E, w)$, the degree constraint k, an integer r, the size constraint s and an aggregation function f, the problem is to find Top-r size-constrained k-Influential Community (TIC) with the highest influence value under the aggregation function f.*

If the size constraint is not emphasized, where we set the size constraint $s = |V|$, the problem would be size-unconstrainted. The following example demonstrates how aggregation functions impact the top-r k-influential community search problem, as well as the difference between size-constrained problem and size-unconstrained problem.

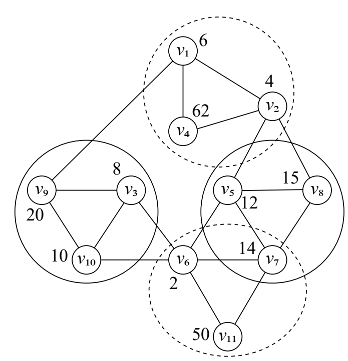


Fig. 1: An example network

Example 1. *As shown in Figure 1, if the aggregation function is sum and $k = 2$, the top-2 k-influential community are $\{v_1, v_2, \ldots, v_{11}\}$ and $\{v_1, v_2, v_4, \ldots, v_{11}\}$. However, when the aggregation function is avg and $k = 2$, the top-2 k-influential community are $\{v_1, v_2, v_4\}$ and $\{v_6, v_7, v_{11}\}$. If we change the aggregation function to min but maintain $k = 2$, then the top-2 k-influential community become $\{v_5, v_7, v_8\}$ and $\{v_3, v_9, v_{10}\}$.*

The above illustrates the difference under different aggregation functions. Following that, we impose size constraint on the subgraph. We set f as sum, $k = 2$, and $s = 4$, then $\{v_3, v_6, v_9, v_{10}\}$ is a size-constrained k-influential community with influence value 40. Although another community, $\{v_1, v_2, \ldots, v_{11}\}$, has a higher influence value 203, it is not retrieved due to the community's size being larger than 4.

To avoid redundancy in results to TIC problem, some works [16], [17] study the top-r non-contained k-influential community without size constraint when the aggregation function is min. The non-containment constraint, on the other hand, does not work well if the aggregation function is not min.

In Figure 1 for instance, we assume that $k = 2$ and the aggregation function $f(\cdot) = avg$. We could obtain that $\{v_6, v_7, v_{11}\}$, $\{v_5, v_6, v_7\}$, and $\{v_5, v_7, v_8\}$ are all k-influential community. The problem is that these communities have overlaps with each other, which is not permitted in certain real-world scenarios. We propose the definition of non-overlapping community search based on this. The definition is given below:

Definition 5 (Non-overlapping)**.** *Given a weighted and undirected graph $G = (V, E, w)$, the degree constraint k, and integer r and an aggregation function f. We suppose that the result of top-r k-influential community search problem is $\{H_1, H_2, \ldots, H_r\}$. For any two communities H_i and H_j, if $H_i \cap H_j = \emptyset$, we refer the result to non-overlapping.*

Example 2. *As shown in Figure 1, we assume that $k = 2$ and the aggregation function $f(\cdot) = \text{avg}$. Following that, we aim to extract top-3 non-overlapping k-influential communities. The results are $\{v_1, v_2, v_4\}$, $\{v_6, v_7, v_{11}\}$, and $\{v_3, v_9, v_{10}\}$. The influence value of each community is 24, 22 and $38/3$, respectively. There is no overlap between any two communities in the result.*

Example 2 is used to illustrate Definition 5. In fact, a non-overlapping constraint could avoid result overlaps. Thus, we give the definition of top-r non-overlapping k-influential community search problem below:

Problem 2. *(Top-r Non-overlapping Size-Constrained k-Influential Community). Given a weighted and undirected graph $G = (V, E, w)$, the degree constraint k, an integer r, the size constraint s, and an aggregation function f, the problem is to find <u>TO</u>p-r <u>N</u>on-overlapping size-constrained k-<u>I</u>nfluential community (TONIC) with the highest influence value under the aggregation function $f(\cdot)$.*

Unfortunately, regardless of whether $f(\cdot) = avg$ or sum, the top-r size-constrained k-influential community search problem is NP-hard. The theoretical analysis would be presented in Section III.

III. Problem Hardness

We provide hardness analysis of top-r k-influential community search problems with or without size constraint in this section. We focus on two aggregation functions: sum and avg, since $f(\cdot) = min$ has been investigated by previous works [16], [17]. Additionally, the algorithms discussed in the preceding studies could simply be extended to the cases when $f(\cdot) = max$.

The hardness of problems can be different with various aggregation functions. When $f(\cdot) = min$ or max, previous works have shown that it could be solved by polynomial-time algorithms for the top-r k-influential community search problem. It is $true$ that the problem could also be solved in polynomial time when $f(\cdot) = sum$. While $f(\cdot) = avg$, the problem is NP-hard. When the size constraint is considered, the top-r k-influential community search problem is NP-hard, no matter what the aggregation function is. The hardness analysis is listed below.

Top-r k-influential Community Search. We provide polynomial-time algorithms for the top-r k-influential community search problem if $f(\cdot) = sum$. The algorithms would be presented in Section IV and we would analyze the correctness of the algorithms. Nonetheless, when $f(\cdot) = avg$, the top-r k-influential community problem is NP-hard. It could not be solved in polynomial time unless P = NP. We demonstrate the hardness of the top-r k-influential community search problem by reducing an NP-complete problem, the decision version of the maximum clique search problem, to our problem. The decision version of the maximum clique search problem is to determine whether a graph G contains a clique of size k.

Theorem 1. *When $f(\cdot) = avg$, the top-r k-influential community search problem is NP-hard.*

Proof. Given a graph $G = (V, E, w)$, we assign each vertex $v_i \in V$ with weight 0. Then, we build another graph $G' = (V', E', w')$ by adding a new vertex u that connects all vertices in V. We set the weight of the new vertex u as w_c. Suppose that there exists a polynomial-time algorithm to address the top-r k-influential community search problem. Then, we could determine whether there exists a (k-1)-clique since the influence value of top-1 k-influential community is $(w_c + k \cdot 0)/(k+1)$ if there exists a (k-1)-clique in graph G. Notably, adding any new vertex (or vertices) into such a clique would only increase the denominator of the influence value. However, the decision version of the maximum clique search problem is NP-complete. It is a contradiction. Thus, the top-r k-influential community search problem is NP-hard, when $f(\cdot) = avg$. □

The objective function is $g(H) = \mathbb{1}_{\delta(H) \geq k} \cdot f(H)$[5] to denote the objective function of the top-r k-influential community search problem, where $\delta(H)$ stands for the minimum degree of the subgraph H and $f(H)$ is the aggregation function. Then we could obtain the following theorems:

[5]Generally, $\mathbb{1}_{\delta(H) \geq k} = 1$ if $\delta(H) \geq k$ holds. Otherwise, $\mathbb{1}_{\delta(H) \geq k} = 0$.

Theorem 2. *If the aggregation function $f(\cdot) = avg$, the objective function $g(\cdot)$ of the k-influential community search problem, is neither submodular nor monotonic.*

Proof. For two arbitrary vertex sets A and B, if $g(\cdot)$ is *submodular*, it must hold that $g(A)+g(B) \geq g(A \cup B)+g(A \cap B)$. We reconsider Figure 1, if $k = 2$, $A = \{v_5\}$ and $B = \{v_6, v_7\}$. Then, $g(A) + g(B) = 0 < g(A \cup B) + g(A \cap B) = 14/3$.

In terms of monotonic analysis, Figure 1 is used as an example again. If $k = 2$, $A = \{v_5\}$ and $B = \{v_5, v_6, v_7\}$, then $g(A) = 0 < g(B) = 14/3$. Nevertheless, if k keeps unchanged, and let $A = \{v_6, v_7, v_8\}$, $B = \{v_5, v_6, v_7, v_8\}$. We could deduce that $g(A) = 7 > g(B) = 22/4$. This indicates that the objective function is not *monotonic*. □

We demonstrated that the top-r k-influential community search problem is NP-hard when $f(\cdot) = avg$. Additionally, we aim to demonstrate that no constant-factor approximated solutions exist for this problem. Before presenting the theoretical analysis, we introduce the Mimimum Subgraph of Minimum Degree $\geq k$ (MSMD$_k$). The objective of MSMD$_k$ problem is to identify a subset H of the vertex set V such that $|H|$ is minimized and $\delta(H) \geq k$.

Theorem 3. *When $f(\cdot) = avg$, there does not exist any constant-factor approximated approaches for the top-r k-influential community search problem.*

Proof. [41] demonstrates that, for $k \geq 3$, the MSMD$_k$ problem does not permit any constant-factor approximation, unless $P = NP$. We show that this is also $true$ for our problem. Given a graph $G = (V, E, w)$, we assign each vertex $v_i \in V$ with weight w_c. Then, a dummy vertex u is added, whose weight $w_u = |V| \cdot w_c$.

Moreover, the vertex u is connected to every vertex of G. Let $\alpha < 1$, if there exists an α-approximated algorithm for top-1 $(k+1)$-influential community search problem, then we could find a $(4/\alpha)$-approximation algorithm for MSMD$_k$ problem. The reason for this is as follows: we use S_{opt} to denote the optimal solution to MSMD$_k$ problem, whereas S^* to denote the approximated solution. Then, according to the definition of average aggregation function, we have

$$\frac{(|S^*| + |V|) \cdot w_c}{|S^*| + 1} \geq \alpha \frac{(|S_{opt}| + |V|) \cdot w_c}{|S_{opt}| + 1}$$

Then, it is obvious that

$$\frac{|S^*|}{|S_{opt}|} \leq 2 \cdot \frac{|S^*| + 1}{|S_{opt}| + 1} \leq \frac{2}{\alpha} \cdot \frac{|S^*| + |V|}{|S_{opt}| + |V|} \leq \frac{2}{\alpha} \cdot \frac{2|V|}{|V|} \leq \frac{4}{\alpha}$$

Thus, there is a contradiction. □

Top-r Size-constrained k-influential Community Search. Since we demonstrated that if $f(\cdot) = avg$, the top-r k-influential community search problem is NP-hard. Then, the top-r size-constrained k-influential community search problem is also NP-hard when $f(\cdot) = avg$. Thus, we focus on the condition that the aggregation function is sum in this part. We reduce the k-clique search problem to the top-r size-constrained k-influential community search problem again to prove that the latter one is NP-hard.

Theorem 4. *Given an aggregation function $f(\cdot) = sum$ and size constraint s, the top-r size-constrained k-influential community search problem is NP-hard.*

Proof. We are given a weighted and undirected graph $G = (V, E, w)$ and $s = k + 1$. If we could solve the top-r size-constrained k-influential community search problem in polynomial-time, then there also exists a polynomial-time solution to k-clique search problem, which is a contradiction. □

IV. Solutions to Size-Unconstrained Problem

In this section, we investigate top-r size-unconstrained k-influential community search problem. We focus primarily on the top-r k-influential community search problem, when $f(\cdot) = sum$. We claim some critical properties of these problems. We utilize sum as an example to illustrate how some pruning techniques are used to accelerate this polynomial-time problem. Moreover, we analyze the time complexity and correctness analysis of the algorithms, respectively. Furthermore, the polynomial-time algorithm could be extended to other aggregation functions.

As for certain circumstances when the problem is NP-hard, we demonstrate the relationship between the size-unconstrained and size-constrained problems. Some heuristic methods are introduced in Section V, which could also be used to address problems with size constraint.

A. Polynomial-Time Problems

To the best of our knowledge, there are two types of top-r k-influential community search problems that could be solved in polynomial-time. The first is a single node which dominates the influence value of the influential community, such as $\min$, $\max$. The second is that the influence value of the influential community is proportional to the number of nodes, such as sum, $sum\text{-}surplus$. We formally define them as follows:

Definition 6 (Node Domination Aggregation Function). *An aggregation function $f(\cdot)$ is called a Node Domination Function if $\forall$ subgraph $H \in G$, $\exists v \in H$, s.t., $f(G[H]) = f(\{v\})$.*

Definition 7 (Size Proportional Aggregation Function). *An aggregation function $f(\cdot)$ is called a Size Proportional Function if for any subgraphs $H, H' \in G$, and $H \subset H'$, then $f(G(H)) \leq f(G(H'))$.*

The first class of problems have been studied in [16], [17], where they propose some strategies for decreasing the search space. We demonstrate that the second type of problem could also be accelerated by using of the properties of aggregation function. Note that it is p-solvable if aggregation functions are monotonic.

We take the $f(\cdot) = sum$ as an example. Given aggregation function sum, the naïve approach to solve the top-r k-influential community search problem is to compute the maximal k-core as well as all the connected components in the maximal k-core. Then, we iteratively visit each vertex in the original graph to check if it is contained in any of the aforementioned connected components. If the answer is $true$, remove the vertex from the connected component it belongs to. We maintain a priority list containing the top-r connected components in each iteration.

In Algorithm 1, L_0 is a set of all disjoint connected components of k-core(G) (Line 1), since a k-core of G could be divided into several disjoint connected components. For all the components, their influential values could be easily computed using sum. After that, L is a set of the top r influential components (Line 2).

Due to the property of sum and all the influence values being nonnegative, we could safely prune a candidate community and all of its subgraphs if it is not in the top r connected components. Thus, we try to remove one vertex from all the top r influential communities (Lines 3-10). Some newly connected communities are generated (Line 8), and we combine them with the current top r influential communities to produce the new top r (Lines 9 and 10).

Algorithm 1: SUM-NAÏVE(G, k, r)

1 $L_0 \leftarrow$ all the disjoint connected components of k-CORE(G, k);
2 $L \leftarrow$ the top-r influential disjoint connected components of L_0;
3 **for** $i \leftarrow 1$ ***to*** $|V|$ **do**
4     $L_c \leftarrow \emptyset$;
5   **for** $j \leftarrow 1$ ***to*** $|L|$ **do**
6   **if** $v_i \in L[j]$ **then**
7             $H \leftarrow L[j] \setminus \{v_i\}$;
8             $C \leftarrow$ all the disjoint connected components of k-CORE(H, k);
9             $L_c \leftarrow L_c \cup C$;
10    $L \leftarrow$ the top-r connected components of $L \cup L_c$;
11 **return** L;

Correctness. We present Corollary 1 and Theorem 5 to demonstrate that Algorithm 1 could output top-r k-influential communities correctly.

Corollary 1. *Each connected component obtained by Algorithm 1 is a k-influential community.*

Theorem 5. *Algorithm 1 correctly identifies the top-r k-influential communities, when $f(\cdot) = sum$, given a graph $G = (V, E, w)$, degree constraint k, and output size constraint r.*

Proof. When $f(\cdot) = sum$, removing any vertex from a community reduces its influence value. Therefore, if a community is not considered as a top-r k-influential community, its subgraphs cannot be regarded as a top-r k-influential community. Furthermore, we could prune this impossible influential community without missing any top-r k-influential community. □

We assume that there exists an H that is a top-r k-influential community, but is not included in the final outcome L. Line 10 implies that there must be at least r influential communities with a greater influence value. As a consequence, we prove the correctness of Algorithm 1.

Complexity. Clearly, Algorithm 1 is a polynomial-time algorithm. We use the notation of n to signify the number of vertices, and m to denote the number of edges in graph G's maximal k-core. Lines 1-2 require $O(n + m)$ time to complete. After that, we have a maximum of n iterations. In each iteration, we delete one vertex from the connected component (Line 3). Additionally, we check whether the vertex is in connected components at each iteration (Line 6), which requires $O(1)$ if using a hash table.

If the connected component contains the vertex, the vertex would be removed first (Lines 5-6). The preceding step takes $O(r)$ time to complete, since we need to check at most r components. Following that, we utilize breadth-first search to determine the connected components of the original component that do not include one of its vertex (Line 8); this takes $O(n+m)$. In conclusion, Algorithm 1 has a time complexity of $O(n \cdot r(n+m))$.

Although the naïve solution is a polynomial-time algorithm, Algorithm 1 is unsatisfactory as the size of the graph increases. Algorithm 1 is inefficient since it requires checking if the component includes the vertex. Additionally, we ignore a key point: the influence value of community is strictly decreased, as shown in Corollary 2. We could integrate some pruning techniques into our algorithms as a result of this. Furthermore, in some cases, we are not required to determine the exact top-r solutions. Thus, we propose an ϵ-approximated algorithm with theoretical guarantees, where ϵ denotes the approximation ratio.

Algorithm 2 illustrates the improved algorithm, while Theorem 6 provides the theoretical analysis. The main procedure of Algorithm 2 is similar to Algorithm 1, but a lower bound LB is used to pruned some unnecessary candidates. To begin, we initialize some variables. Among them, L_0 and L are the same as those in Algorithm 1. Notably, a lower bound LB is defined as $(1 - \epsilon) \times f(L_{max})$, where ϵ is a predefined parameter[6] and $f(L_{\max})$ is the maximum influence value among all the disjoint connected components of L_0. In the While-loop (Lines 7-19), we set L_{max} as the community with the largest influence value in L (Line 8), and LB as the new lower bound as $(1-\epsilon) \times$ the largest influence value (Line 9). Then, we try to remove one vertex of the L_{max} to produce the new candidates (Lines 11-19). The main reason is that the L_{max} has a high probability to produce new candidates with an influence value larger than LB. Two pruning rules are used for the new candidates (Lines 13 and 16). After that, the new candidates are added to the L (Line 18) and then set L as the top r influential communities (Line 19).

Correctness. We analyze the correctness of Algorithm 2, where Corollary 2 and Theorem 6 are presented below.

Corollary 2. *If we remove any vertices from the influential community, the influence value of the k-influential community would decrease.*

Remark 1. *If f does not satisfy Corollary 2, then Algorithm 2 could not cope with it. Nevertheless, we could revise the corresponding function in the local search to solve it.*

[6] $\epsilon = 0.1$ by default in this paper.

Algorithm 2: TIC-IMPROVED(G, k, r, $f(\cdot)$, ϵ)

1 $L_0 \leftarrow$ compute the disjoint connected components of maximal k-core of G;
2 $L \leftarrow$ the top-r disjoint connected components of L_0;
3 $L_r \leftarrow$ the r-th largest influence value community in L;
4 $L_{\max} \leftarrow$ the community with largest influence value in L;
5 $LB \leftarrow f(L_{\max}) \times (1-\epsilon)$;
6 $R \leftarrow$ communities in L with influence value $\geq LB$;
7 **while** $|R| < r$ **do**
8     $L_{\max} \leftarrow$ the community with largest influence value in L;
9     $LB \leftarrow f(L_{\max}) \times (1-\epsilon)$;
10     $L \leftarrow L \setminus L_{\max}$;
11   **for** $v \in L_{max}$ **do**
12         $H \leftarrow L_{max} \setminus \{v\}$;
13   **if** $f(H) > f(L_r)$ **then**
14             $C \leftarrow$ compute the connected k-core of H;
15   **for** $i = 1$ ***to*** $|C|$ **do**
16   **if** $f(C[i]) \geq LB$ **then**
17                     $R \leftarrow R \cup C[i]$;
18             $L \leftarrow L \cup C$;
19             $L \leftarrow$ the top-r connected components of L;
20 **return** R;

According to Lemma 2 of [42]–[44], we define the approximation factor:

Definition 8 (Approximation Factor)**.** *If the influential values for the exact top r result are $\{v_1, v_2, ..., v_r\}$, a result set $\{iv_1, iv_2, ..., iv_r\}$ is defined as a $1-\epsilon$ approximation if*

$$iv_r \geq (1-\epsilon)v_r \tag{1}$$

Theorem 6. *Given a weighted and undirected graph $G = (V, E, w)$, a degree constraint k, an output size constraint r, and an approximation ratio ϵ, we use r_e to denote the influence value of the r-th influential community of the output of the exact algorithm, and r_a to denote the influence value of the r-th largest influence value community of the output of the Algorithm 2. If $f(\cdot) = sum$, we could obtain that $r_a/r_e \geq 1-\epsilon$.*

Proof. At each iteration, we choose the maximal influential community from the candidate communities list. The maximal influential community is larger or equal to r-th largest influence community in terms of influence value. Thus, $r_a/r_e \geq 1-\epsilon$. □

Complexity. Algorithm 2 is straightforward and efficient. We would instantly produce a k-influential community whose influence value exceeds the lower bound. The time complexity of Algorithm 2 is $O(rn(m+n))$. Additionally, it takes $O(n+m)$ time in Line 1. However, in Line 7, we simply calculate r iterations, and the time complexity of Lines 11-19 is $O(r(n+m))$. To put it in a nutshell, the time complexity is $O(rn(n+m))$.

Non-overlapping. when $f(\cdot) = sum$ and the community is size-unconstrained, we merely execute Lines 1-3 of Algorithm 2 to compute the top-r non-overlapping k-influential community. This is due to the fact that we would obtain the community with the largest influence each time. After obtaining it, we would remove it from the graph. As a consequence, we could obtain the correct result by performing Lines 1-3 of Algorithm 2.

Discussion. If $f(\cdot) \neq sum$, the preceding algorithm could potentially be expanded to solve other top-r k-influential communities. For instance, $f(\cdot) = sum\text{-}surplus$ also satisfies Corollary 2. Thus, we could use Algorithm 2 to solve the top-r k-influential community search problem for $sum\text{-}surplus$.

Power-Law Graph. In practice, the degree distribution of the graph conforms to the power-law distribution. Thus, it is critical to analyze the complexity of our algorithm under power-law distribution.

Definition 9 (Power-Law Graph)**.** *Given a graph $G = (V, E)$, the degree distribution of the graph follows a power-law distribution, if the fraction $P(k)$ of nodes in the graph having k connections to other nodes goes for large values of k as $P(k) \sim k^{-\gamma}$, where $2 < \gamma < 3$.*

Lemma 1. *When we consider the power-law graph, the number of nodes with a degree greater or equal to k is $n/((\gamma-1)k^{\gamma-1})$, and the number of edges is bounded by $n/(2(\gamma-2)k^{\gamma-2})$.*

Proof. According to the definition of the power-law graph, the number of node with a degree larger or equal to k is $n/((\gamma-1)k^{\gamma-1})$. Then, the number of nodes whose degree is equal to k is n/k^{γ}. Thus, the total number of edges is bounded by

$$\frac{n}{2}\sum_{d=k}^{\infty}\frac{1}{d^{\gamma-1}} \leq \frac{n}{2}\int_{k}^{\infty} x^{-\gamma+1}dx \leq \frac{n}{2(\gamma-2)k^{\gamma-2}}$$

This completes the proof. □

According to the definition of a power-law graph, then the time complexity of our algorithm under a power-law graph could be $O(\frac{rn^2((k+2)\gamma-k-4)}{2(\gamma-1)^2(\gamma-2)k^{(2\gamma-2)}})$, when the degree constraint is k, where $2 < \gamma < 3$.

B. NP-Hardness

We investigate the TIC problem under several aggregation functions when it could be solved in polynomial time. Nevertheless, the problem would be NP-hard with some aggregation functions. In Section II, we demonstrate this problem's hardness analysis. As a result, heuristic algorithms must be developed to address the NP-hard problem.

V. SOLUTIONS TO SIZE-CONSTRAINED PROBLEM

Section II demonstrates that when the size of an influential community is constrained, the TIC problem is NP-hard. In this section, we provide the exact algorithms for the TIC problem. Given that the top-r size-constrained is NP-hard, we present a heuristic algorithm to address it in polynomial time. Note that all the techniques could be easily extended to the TONIC problem.

A. Exact Algorithm

The exact algorithm is quite time-consuming, and the naïve exact algorithm for the TIC problem is illustrated in Algorithm 3. The key idea is to enumerate all possible solutions and return the top-r k-influential community. In Algorithm 3, we set the candidate community set as $\emptyset$ (Line 1). Then, a for-loop (Lines 2-5) enumerates all possible communities whose size ranges from $k+1$ to s (Line 3). Then, the k-core and connected constraints are verified in Line 4. The newly added candidates would be inserted into L (Line 5), and then the top-r influential connected components would be returned (Line 6).

Algorithm 3: TIC-EXACT(G, k, r, s, $f(\cdot)$)

1 $L \leftarrow \emptyset$;
2 **for** $i = k+1$ ***to*** s **do**
3 $\quad$ $L_c \leftarrow$ enumerate all possible vertex sets whose size is i;
4 $\quad$ **for** $c \in L_c$ *and* c *is a connected* k*-core* **do**
5 $\quad\quad$ $L \leftarrow L \cup c$;
6 **return** top-r connected components of L;

Time Complexity. There are $\sum_{i=k+1}^{s} C_n^i$ possible vertex sets. For each of them, Line 4 takes $O(m+n)$ to verify the k-core and connected constraints. Thus, the time complexity of naïve exact algorithm is $O(\sum_{i=k+1}^{s} C_n^i \cdot (m+n))$. To tackle this issue, we provide efficient heuristic solutions to the top-r size-constrained k-influential community search problem, namely *Local Search*.

B. Local Search Algorithm

The local search algorithm is to begin with each vertex u in the graph, and then search the s nearest neighbors of u. Following that, we determine whether u could construct k-core in the absence of its neighbors. Algorithm 4 summarizes the approach. In Algorithm 4, L is the set of disjoint connected components after the k-core(G) (Line 1). Then, a for-loop explores every vertex $v \in V$ (Lines 2-7). The s-nearest neighbors of v_i would be assigned to V_i (Line 4), which would be verified in the Strategy Procedure according to various $f(\cdot)$ (Line 7). Notably, if v_i does not have s neighbors, we would explore its 2-hop neighbors. Thus, a BFS could be conducted in Line 4. If the $greedy$ signal is set, the vertices in V_i would be sorted in the descending order of influence value (Lines 5-6). Line 7 would add new candidates to L. After that, the top r influential communities of L would be sorted and returned (Lines 8-9).

With different aggregation functions, we would use different strategies in Algorithm 4 Line 7. We present two strategies to illustrate how to design heuristic strategies with various aggregation functions. As for sum, the first procedure is used to tackle the top-r size-constrained k-influential community problem. In $SumStrategy$, the candidate set C is set $\emptyset$ and L_r is set as the r-th largest influential community in Lines 1 and 2, respectively. A while-loop (Lines 3-5) selects the first s vertices as a candidate community. Then, a while-loop (Lines 6-12) compute the connected k-core using the vertices

Algorithm 4: LOCALSEARCH(G, k, r, $greedy$, s, $f(\cdot)$)

1 $L \leftarrow$ Compute the maximal k-core of G;
2 **for** $i = 1$ ***to*** $|V|$ **do**
3 $\quad$ **if** v_i *is not removed* **then**
4 $\quad\quad$ $V_i \leftarrow$ the vertex set of an s-nearest neighbor of v_i;
5 $\quad\quad$ **if** $greedy = True$ **then**
6 $\quad\quad\quad$ Sort the vertices in V_i in the descending order of influence value;
7 $\quad\quad$ $L \leftarrow$ STRATEGY(V_i, L, gs, f);
8 Sort L by the influence value of each element;
9 **return** L;

in C and update (Line 8) the result influential community L if $f(C) > f(L_r)$.

Remark 2. *The effects of local search heuristic depend on the diameter of the result community. If small, local search works wells since local search preferentially search the closer candidates.*

Procedure SumStrategy(V, L, f)

1 $C \leftarrow \emptyset$;
2 $L_r \leftarrow$ the r-th largest influence value community;
3 **while** $|C| < s$ **do**
4 $\quad$ $C \leftarrow C \cup V.front$, $V \leftarrow V \setminus V.front$;
5 $\quad\quad$ $\triangleright$ $V.front$ is the first element of V
6 **while** $|C| > k$ *and* $f(C) > f(L_r)$ **do**
7 $\quad$ **if** C *is* k*-core* **then**
8 $\quad\quad$ $L \leftarrow L \setminus L_r$, $L \leftarrow L \cup C$;
9 $\quad\quad$ **break**;
10 $\quad$ **else**
11 $\quad\quad$ $C \leftarrow C \setminus C.last$;
12 $\quad\quad\quad$ $\triangleright$ $C.last$ is the last element of C
13 **return** L;

Time Complexity. In Algorithm 4, we compute the maximal k-core of graph G (Line 1) in $O(n+m)$ time. Then, in Line 2, it computes n iterations. Line 4 computes the k-nearest neighbor of v_i in $O(k)$ time. If the greedy signal is $true$, then the vertices are sorted, which takes $O(s \log s)$. Moreover, the Procedure SumStrategy requires $O(s^2)$. Thus, if utilizing the greedy strategy, the time complexity of Algorithm 4 is $O(nks^3 \log s)$. Otherwise, the time complexity of Algorithm 4 is $O(nks^3)$.

Nonetheless, if $f(\cdot) = avg$, we will modify the procedure by using its characteristics. The Procedure AvgStrategy concludes the strategy. In Procedure AvgStrategy, we initialize candidate vertex set C and influential candidate community set L_c as $\emptyset$ (Line 1). Line 2 sets L_r as the top r-th largest influential community. A while-loop (Lines 3-10) tries to add vertex $\in V$ to C and test if C could be a new candidate community to L.

If the $greedy$ signal is used (Line 6), we could safely prune L_r from L (Line 7) and add C to L, since the influence value of the latter vertex $\in V$ is no larger than that of the current one. Otherwise, we add C to L_c as a candidate (Line 10). Then, we choose the $L_c[i]$ with the maximum influence value among L_c, and add it to L. The time complexity of Algorithm 4 is the same as the analysis outlined above.

Non-overlapping. The objective is to compute the top-r size-constrained non-overlapping k-influential community. As a result, we could slightly modify the Local Search Algorithm (Algorithm 4). In Line 7, when designing strategy, we could remove each k-influential community once it is obtained by a local search algorithm.

Procedure AvgStrategy(V, L, greedy, f)

---

1 $C \leftarrow \emptyset$, $L_c \leftarrow \emptyset$;
2 $L_r \leftarrow$ the r-th largest influence value community;
3 **while** $|C| < s$ **do**
4 $\quad C \leftarrow C \cup V.front$, $V \leftarrow V \setminus V.front$;
5 $\quad$ **if** $|C| > k$, $f(C) > f(L_r)$ *and* C *is* k*-core* **then**
6 $\quad\quad$ **if** $greedy = True$ **then**
7 $\quad\quad\quad L \leftarrow L \setminus L_r$, $L \leftarrow L \cup C$;
8 $\quad\quad\quad$ **break**;
9 $\quad\quad$ **else**
10 $\quad\quad\quad L_c \leftarrow L_c \cup C$;
11 **if** $greedy = False$ **then**
12 $\quad L_{max} \leftarrow \underset{i \in \{1,2,\ldots,|L_c|\}}{\arg\max} L_c[i]$;
13 $\quad L \leftarrow L \setminus L_r$, $L \leftarrow L \cup L_{\max}$;
14 **return** L;

VI. EXPERIMENTS

In this section, we conduct extensive experiments on our proposed technique. All algorithms are implemented in C++. All experiments are conducted on a Linux machine with an Intel Xeon 2.70GHz CPU and 400GB memory.

TABLE III: Datasets

Dataset	#vertices	#edges	$d_{\max}$	d_{avg}	$k_{\max}$
DomainPub	$22,692$	$60,830$	125	5.35	31
Email	$36,692$	$183,831$	$1,383$	10.02	43
DBLP	$317,080$	$1,049,866$	343	6.62	113
Youtube	$1,134,890$	$2,987,624$	$28,754$	5.27	51
Orkut	$3,072,441$	$117,185,083$	$33,313$	76.28	253
LiveJournal	$3,997,962$	$34,681,189$	$14,815$	17.35	360
FriendSter	$65,608,366$	$1,806,067,135$	$5,214$	55.06	304

Datasets. We evaluate our experiments on 6 real graphs: Email, DBLP, Youtube, Orkut, LiveJournal, and FriendSter. All datasets are downloaded from the Stanford Network Analysis Platform[7]. The statistics of datasets are shown in Table III. In Table III, $k_{\max}$ indicates that graph does not contain a non-empty $(k_{\max}+1)$-core. Moreover, the weight of vertices is the PageRank value of vertices with the damping factor being set as 0.85. In order to demonstrate the effectiveness of the proposed algorithms, we illustrate the case study over Aminer dataset.

Compared Algorithms. To the best of our knowledge, there only exist algorithms for the top-r k-influential community search problem when $f(\cdot) = min$. The algorithm is simply extended to the case when the aggregation function is max. However, there exists no work studying the top-r k-influential community search problem under other aggregation functions, e.g., avg, sum. Their algorithms were designed based on the nice property of min. Thus, if simply modify them to our problem, they would degrade to our naïve algorithm. We do not consider these algorithms in this paper.

Due to the slowness of the exact algorithm, we implement and evaluate the following approaches.

Size-Unconstrained Problem:

- *Naïve*: The solution to top-r size-unconstrained k-influential community search problem for sum, which is proposed in Algorithm 1.
- *Improve*: The method mentioned in Algorithm 2. We set ϵ equal to 0.
- *Approx*: The method mentioned in Algorithm 2 when the $\epsilon > 0$.

Size-Constrained Problem:

- *Random*: The method mentioned in Algorithm 4 by using random strategy. The random approach does not sort the nearest neighbor set.
- *Greedy*: The method mentioned in Algorithm 4 by using greedy strategy. The greedy approach sorts the nearest neighbor set in decreasing order.

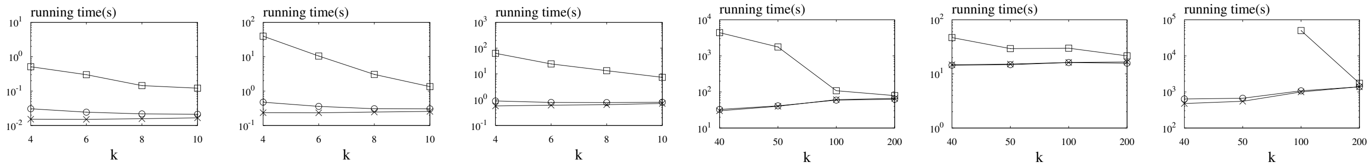
Parameters. There are four parameters that need to be considered: k, r, ϵ, and s. We conduct experiments under various settings by varying four parameters. The value of k varies from 4 to 200, r varies from $\{5, 10, 15, 20\}$, ϵ varies from $\{0.01, 0.05, 0.1, 0.2\}$, and s varies from $\{5, 10, 15, 20\}$. The default setting is $\epsilon = 0.1$, $r = 5$, and $s = 20$. However, since the naïve algorithm is inefficient, we will default to $k = 4$ for small datasets, e.g., Email, DBLP, Youtube, and $k = 40$ for large datasets, e.g., Orkut, LiveJournal, FriendSter. By default, we evaluate the performance of overlapping approaches. In the case study, we examine non-overlapping approaches to demonstrate the difference between our models and previous work.

A. Approaches to Size-Unconstrained Problems

Exp-I: Effect of k. Figure 2 evaluates the performance of three methods by fixing $r = 5$, and varying k (missing point indicates the algorithm cannot terminate in one day). What is striking in this figure is that as k increases, the running time of the naïve algorithm decreases.

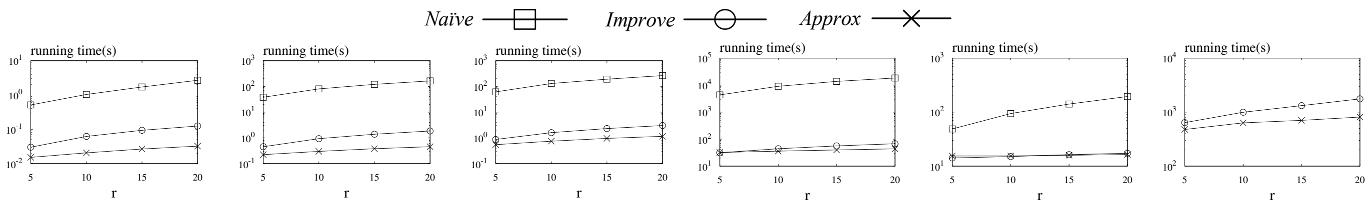
The main reason is that because the graph size decreases after pruning, and the running time of the core-decomposition algorithm is a tiny fraction of the overall time of the naïve algorithm. Nevertheless, the running time of the improved algorithm and approximated algorithms both grow as k increases, and the running time of two algorithms is comparable.

[7]http://snap.stanford.edu/



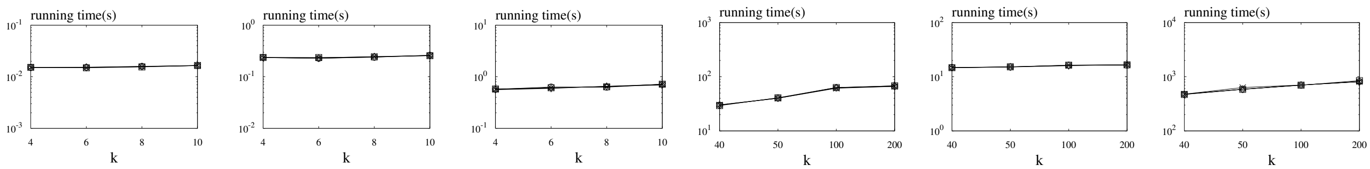
(a) Email (b) DBLP (c) Youtube (d) Orkut (e) Livejournal (f) FriendSter

Fig. 2: Running time vs. k (sum, size-unconstrained)



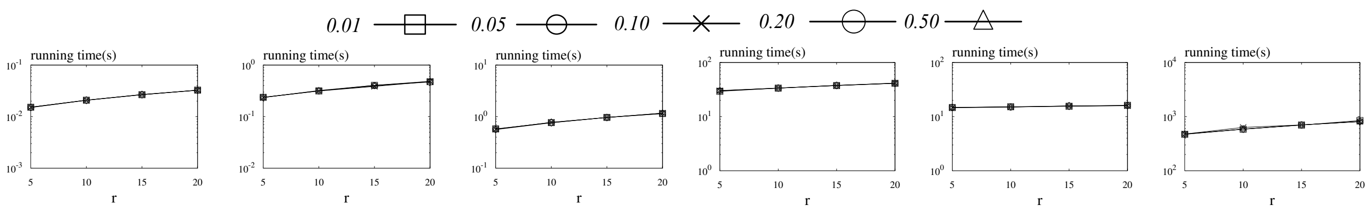
(a) Email (b) DBLP (c) Youtube (d) Orkut (e) Livejournal (f) FriendSter

Fig. 3: Running time vs. r (sum, size-unconstrained)



(a) Email (b) DBLP (c) Youtube (d) Orkut (e) Livejournal (f) FriendSter

Fig. 4: Running time vs. k (ϵ, sum, size-unconstrained)



(a) Email (b) DBLP (c) Youtube (d) Orkut (e) Livejournal (f) FriendSter

Fig. 5: Running time vs. r (ϵ, sum, size-unconstrained)

The reason for this is that the core-decomposition consumes a significant part of running time.

Exp-II: Effect of r. The effect of r is evaluated in this experiment. As demonstrated in Figure 3, when r increases, the running time of algorithms increases. The reason is that it needs to output more k-influential communities and the iterations of the algorithm increases.

Exp-III: Impact of ϵ. We compare the approximated algorithms to the top-r k-influential community search problem under different aggregation functions. ϵ ranges from $\{0.01, 0.05, 0.1, 0.2, 0.5\}$. The result is demonstrated in Figure 4 and Figure 5. What stands out in these figures is the running time of approximated algorithm remains almost unaltered by varying ϵ. As a result, the approximated algorithm is insensitive to ϵ. The reason is that the top-r k-influential community is always computed within r iterations at the beginning.

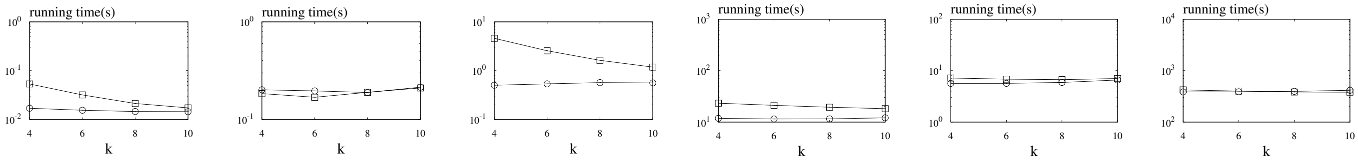
B. Approaches to Size-constrained Problems

Exp-IV: Effect of k. In this exp, we vary k to evaluate the Local Search technique. Closer inspection of Figures 6 and 7 shows that the running time of algorithms decreases because the size of the graph decreases as k increases. As the size of the graph decreases, the iteration of the local algorithm decreases as well.

Exp-V: Effect of r. We vary r to evaluate the Local Search algorithm. Figure 8 and Figure 9 demonstrate that the performance of the Local Search algorithm is insensitive to r. Notably, r would not be very large in reality since it is infeasible for uses to choose from numerous candidates. In such a scenario, the Local Search algorithm is insensitive to r, since the algorithm would always compute more than r k-influential communities. Thus, when r is not large, its value would not affect the performance of the algorithm.

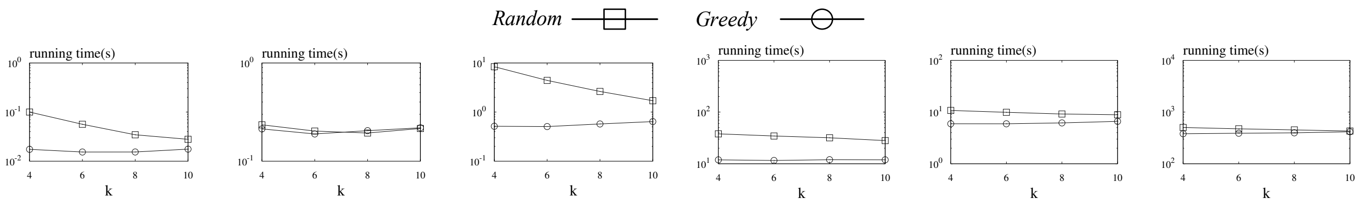
Exp-VI: Effect of s. In this experiment, we vary s to evaluate the efficiency of the Local Search algorithm. Figures 10 and 11 indicate that the running time of algorithms increases since we have to search more neighbor vertices at each iteration with the increase of s.

Exp-VII: Effectiveness. In this evaluation, we compare the greedy strategy with the random one. We fix $r = 5$ and $s = 20$, by varying k from $\{4, 6, 8, 10\}$. Figure 12 and Figure 13 demonstrate that the influence value of r-th k-influential community obtained by greedy strategy is always larger than that computed by random strategy. This is because the size of the community is constrained. If we select the vertex with the largest influence value, a community with a larger influence value could be produced.



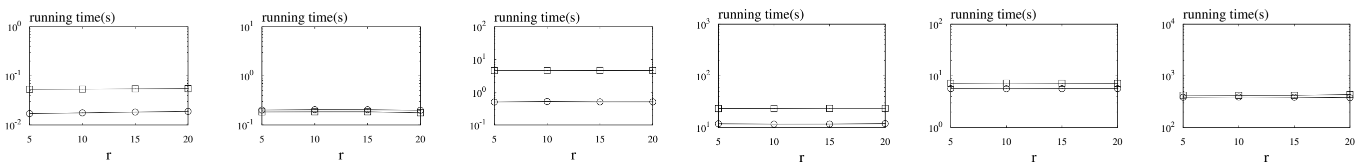
(a) Email (b) DBLP (c) Youtube (d) Orkut (e) Livejournal (f) FriendSter

Fig. 6: Running time vs. k (sum, size-constrained)



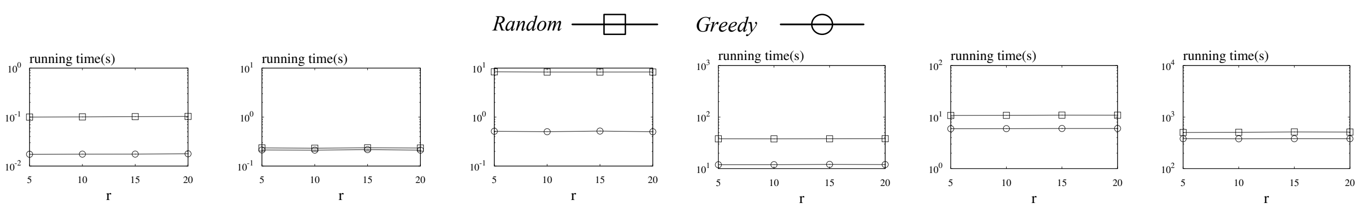
(a) Email (b) DBLP (c) Youtube (d) Orkut (e) Livejournal (f) FriendSter

Fig. 7: Running time vs. k (avg, size-constrained)



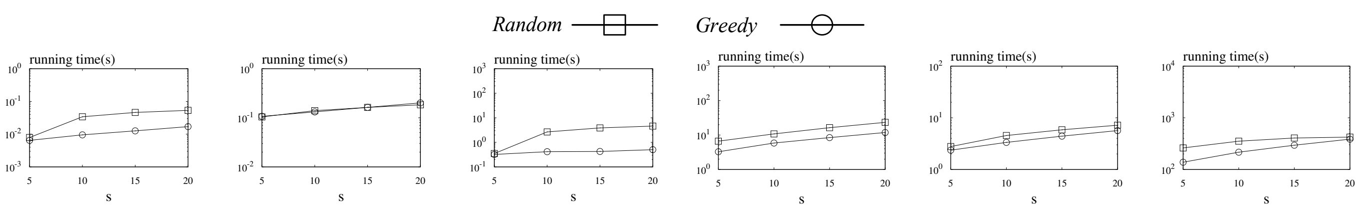
(a) Email (b) DBLP (c) Youtube (d) Orkut (e) Livejournal (f) FriendSter

Fig. 8: Running time vs. r (sum, size-constrained)



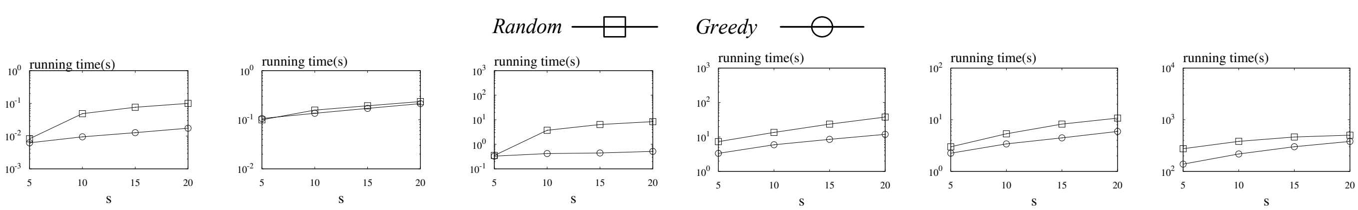
(a) Email (b) DBLP (c) Youtube (d) Orkut (e) Livejournal (f) FriendSter

Fig. 9: Running time vs. r (avg, size-constrained)



(a) Email (b) DBLP (c) Youtube (d) Orkut (e) Livejournal (f) FriendSter

Fig. 10: Running time vs. s (sum, size-constrained)



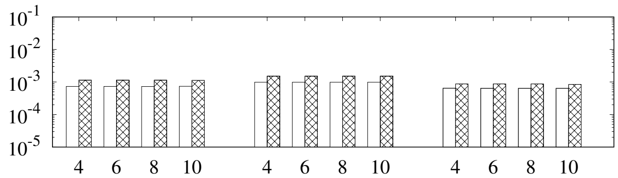
(a) Email (b) DBLP (c) Youtube (d) Orkut (e) Livejournal (f) FriendSter

Fig. 11: Running time vs. s (avg, size-constrained)

C. *Case study*

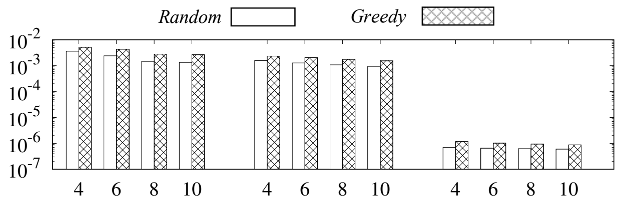
We evaluate a case study for the k-influential community under various aggregation functions on a social network. The dataset could be downloaded from Aminer[8], which is collected for the purpose of cross-domain recommendation. It includes five fields: Data Mining, Medical Informatics, Theory, Visualization, and Database. Each vertex represents a researcher, and the edge between two vertices indicates that they have co-authored at least 1 publication. Figure 14 shows the top-3 non-overlapping k-influential community under different aggregation functions, when $k = 4$. Since our algorithms are

[8]https://www.aminer.org/data



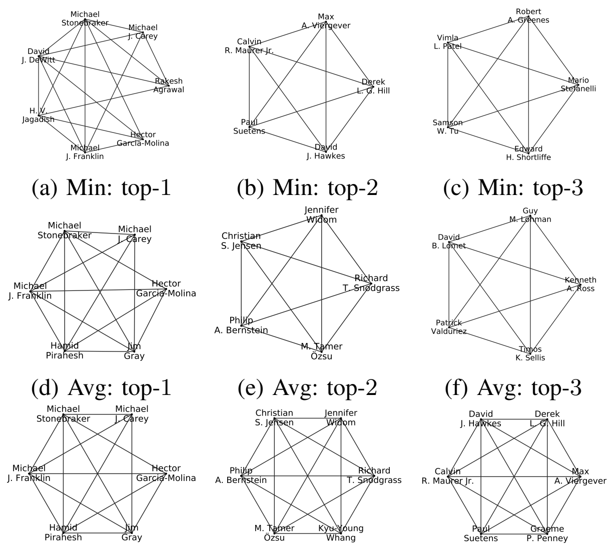
(a) DBLP (b) Orkut (c) LiveJournal

Fig. 12: r-th influence value (sum, size-constrained)



(a) Email (b) Youtube (c) FriendSter

Fig. 13: r-th influence value (avg, size-constrained)



(a) Min: top-1 (b) Min: top-2 (c) Min: top-3

(d) Avg: top-1 (e) Avg: top-2 (f) Avg: top-3

(g) Sum: top-1 (h) Sum: top-2 (i) Sum: top-3

Fig. 14: Case Study: Aminer

heuristic when the aggregation function is sum or avg for top-r size-constrained k-influential community search problem. Thus, the result is not explicit. However, the result of the top-r non-overlapping size-constrained k-influential community under aggregation functions, e.g., sum, avg, could satisfy different requirements in practice.

VII. Related Work

Community Detection. Community detection has been studied for several decades. The goal of community detection is to retrieve all communities that fulfill constraints. It is first investigated in [45], [46]. Following that, many improved methods [47] are proposed. Recently, some researchers study community detection over different kinds of graphs. Some investigate community detection by using machine learning techniques. For instance, Jian et al. [48] present solutions about community detection over heterogeneous networks. Li et al. [49] investigate community detection in the presence of adversarial attacks.

Community Search. Community search has been widely studied by a large number of researchers [50]–[52]. Sozio et al. [53] present a linear-time algorithm to find a maximal connected k-core that contains the set of query vertices. Next, Cui et al. [54] provide more efficient algorithms for the above-mentioned problem. Recently, some scientists have concentrated their efforts on community search over various graphs or with various community models. For example, Fang et al. solve community search over spatial graphs in [55].

Moreover, Fang et al. [56] propose effective and efficient algorithms for community search over heterogeneous graphs. Huang et al. [30], Liu et al. [57] and Chen et al. [58] study community search based on the k-truss community model. As for the influential community search problem, Li et al. [16] firstly study the top-r influential community search problem. Then, an improved online algorithm and a novel progressive method is proposed in [17]. Both of them, however, ignore an essential point: in some cases, the aggregation function is not min, and existed technique cannot be applied directly.

Cohesive Subgraph Mining. Cohesive subgraphs discovery is a practical and fascinating problem in graph mining. There are some definitions to measure cohesive subgraphs. Among them, the maximal clique [59], [60], the k-core [18], [61], [62], the k-truss [20], and the k-edge connected subgraphs [63], [64] are widely-used models. Due to the widespread use of cohesive subgraphs in graph mining, an increasing number of people have focused their attention on this problem in recent years. To illustrate, k-core decomposition is studied in [65], [66] and k-truss decomposition is investigated in [67], [68].

VIII. Future Works

As for the size unconstrained problems that are NP-hard, there is no algorithms proposed. The main obstacle of this problem is the costly search space. For such a problem, a possible direction would be carefully design pruning rules and investigate approximation method. To speedup this process, a parallel or distributed context could also be investigated.

IX. Conclusions

In this paper, we investigate the problem of extracting the top-r k-influential communities in social networks under various aggregation functions. As for the top-r k-influential community search problem, if the aggregation function is sum, we propose an efficient algorithm. Furthermore, we prove the hardness of the problem under size constraint and provide some heuristic algorithms for the top-r size-constraint k-influential community search problem. Finally, extensive experiments on 6 real-world graphs indicate that our algorithms are efficient and effective. The case study reveals that our model has broad applications.

Acknowledgment

This work is supported by Hong Kong RGC ECS grant (No. 24203419), RGC CRF grant (No. C4158-20G), Hong Kong ITC ITF grant (No. MRP/071/20X), and NSFC grant (No. U1936205). It is also supported by the Research Gants Council of Hong Kong, China under No. 14203618, No. 14202919 and No. 14205520.

REFERENCES

[1] Y. Fang, X. Huang, L. Qin, Y. Zhang, W. Zhang, R. Cheng, and X. Lin, "A survey of community search over big graphs," *The VLDB Journal*, vol. 29, no. 1, pp. 353–392, 2020.

[2] Y. Fang, R. Cheng, S. Luo, and J. Hu, "Effective community search for large attributed graphs," *PVLDB*, vol. 9, no. 12, pp. 1233–1244, 2016.

[3] Y. Fang, R. Cheng, X. Li, S. Luo, and J. Hu, "Effective community search over large spatial graphs," *PVLDB*, vol. 10, no. 6, pp. 709–720, 2017.

[4] S. Bian, Q. Guo, S. Wang, and J. X. Yu, "Efficient algorithms for budgeted influence maximization on massive social networks," *Proceedings of the VLDB Endowment*, vol. 13, no. 9, pp. 1498–1510, 2020.

[5] K. Hao, L. Yuan, and W. Zhang, "Distributed hop-constrained st simple path enumeration at billion scale," *Proceedings of the VLDB Endowment*, vol. 15, no. 2, pp. 169–182, 2021.

[6] Z. Yang, L. Lai, X. Lin, K. Hao, and W. Zhang, "Huge: An efficient and scalable subgraph enumeration system," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 2049–2062.

[7] Y. Peng, Y. Zhang, X. Lin, W. Zhang, L. Qin, and J. Zhou, "Towards bridging theory and practice: hop-constrained st simple path enumeration," *Proceedings of the VLDB Endowment*, vol. 13, no. 4, pp. 463–476, 2019.

[8] Y. Peng, Y. Zhang, X. Lin, L. Qin, and W. Zhang, "Answering billion-scale label-constrained reachability queries within microsecond," *Proceedings of the VLDB Endowment*, vol. 13, no. 6, pp. 812–825, 2020.

[9] Y. Peng, X. Lin, Y. Zhang, W. Zhang, and L. Qin, "Answering reachability and k-reach queries on large graphs with label-constraints," *The VLDB Journal*, pp. 1–25, 2021.

[10] Y. Peng, W. Zhao, W. Zhang, X. Lin, and Y. Zhang, "Dlq: A system for label-constrained reachability queries on dynamic graphs," in *Proceedings of the 230th ACM International Conference on Information & Knowledge Management*, 2021.

[11] J. Li, X. Wang, K. Deng, X. Yang, T. Sellis, and J. X. Yu, "Most influential community search over large social networks," in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*. IEEE, 2017, pp. 871–882.

[12] Y. Fang, K. Yu, R. Cheng, L. V. Lakshmanan, and X. Lin, "Efficient algorithms for densest subgraph discovery," *PVLDB*, vol. 12, no. 11, pp. 1719–1732, 2019.

[13] S. Brohee and J. Van Helden, "Evaluation of clustering algorithms for protein-protein interaction networks," *BMC bioinformatics*, vol. 7, no. 1, p. 488, 2006.

[14] X. Du, R. Jin, L. Ding, V. E. Lee, and J. H. Thornton Jr, "Migration motif: a spatial-temporal pattern mining approach for financial markets," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 1135–1144.

[15] X. Qiu, W. Cen, Z. Qian, Y. Peng, Y. Zhang, X. Lin, and J. Zhou, "Real-time constrained cycle detection in large dynamic graphs," *Proceedings of the VLDB Endowment*, vol. 11, no. 12, pp. 1876–1888, 2018.

[16] R.-H. Li, L. Qin, J. X. Yu, and R. Mao, "Influential community search in large networks," *Proceedings of the VLDB Endowment*, vol. 8, no. 5, pp. 509–520, 2015.

[17] F. Bi, L. Chang, X. Lin, and W. Zhang, "An optimal and progressive approach to online search of top-k influential communities," *Proceedings of the VLDB Endowment*, vol. 11, no. 9, pp. 1056–1068, 2018.

[18] S. B. Seidman, "Network structure and minimum degree," *Social networks*, vol. 5, no. 3, pp. 269–287, 1983.

[19] H. Wu, J. Cheng, Y. Lu, Y. Ke, Y. Huang, D. Yan, and H. Wu, "Core decomposition in large temporal graphs," in *2015 IEEE International Conference on Big Data (Big Data)*. IEEE, 2015, pp. 649–658.

[20] J. Cohen, "Trusses: Cohesive subgraphs for social network analysis," *National security agency technical report*, vol. 16, pp. 3–29, 2008.

[21] F. Zhang, Y. Zhang, L. Qin, W. Zhang, and X. Lin, "When engagement meets similarity: efficient (k, r)-core computation on social networks," *arXiv preprint arXiv:1611.03254*, 2016.

[22] R. H. Chitnis, F. V. Fomin, and P. A. Golovach, "Preventing unraveling in social networks gets harder," in *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[23] F. D. Malliaros and M. Vazirgiannis, "To stay or not to stay: modeling engagement dynamics in social graphs," in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 2013, pp. 469–478.

[24] S. Wu, A. Das Sarma, A. Fabrikant, S. Lattanzi, and A. Tomkins, "Arrival and departure dynamics in social networks," in *Proceedings of the sixth ACM international conference on Web search and data mining*, 2013, pp. 233–242.

[25] S. Amer-Yahia, S. B. Roy, A. Chawlat, G. Das, and C. Yu, "Group recommendation: Semantics and efficiency," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 754–765, 2009.

[26] D. Cao, X. He, L. Miao, Y. An, C. Yang, and R. Hong, "Attentive group recommendation," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018, pp. 645–654.

[27] J. K. Kim, H. K. Kim, H. Y. Oh, and Y. U. Ryu, "A group recommendation system for online communities," *International Journal of Information Management*, vol. 30, no. 3, pp. 212–219, 2010.

[28] X. Huang, H. Cheng, L. Qin, W. Tian, and J. X. Yu, "Querying k-truss community in large and dynamic graphs," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 2014, pp. 1311–1322.

[29] K. Yao and L. Chang, "Efficient size-bounded community search over large networks." *Proc. VLDB Endow.*, vol. 14, no. 8, pp. 1441–1453, 2021.

[30] X. Huang and L. V. Lakshmanan, "Attribute-driven community search," *Proceedings of the VLDB Endowment*, vol. 10, no. 9, pp. 949–960, 2017.

[31] X. Chen, Y. Peng, S. Wang, and J. X. Yu, "Dlcr : Efficient indexing for label-constrained reachability queries on large dynamic graphs," *Proceedings of the VLDB Endowment*, 2022.

[32] Q. Feng, Y. Peng, W. Zhang, Y. Zhang, and X. Lin, "Towards real-time counting shortest cycles on dynamic graphs: A hub labeling approach," in *ICDE*. IEEE, 2022.

[33] Z. Yuan, Y. Peng, P. Cheng, L. Han, X. Lin, L. Chen, and W. Zhang, "Efficient k-clique listing with set intersection speedup," in *ICDE*. IEEE, 2022.

[34] B. Saha, A. Hoch, S. Khuller, L. Raschid, and X.-N. Zhang, "Dense subgraphs with restrictions and applications to gene annotation graphs," in *Annual International Conference on Research in Computational Molecular Biology*. Springer, 2010, pp. 456–472.

[35] C. Tsourakakis, F. Bonchi, A. Gionis, F. Gullo, and M. Tsiarli, "Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 104–112.

[36] M. Sozio and A. Gionis, "The community-search problem and how to plan a successful cocktail party," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2010, pp. 939–948.

[37] F. Luo, J. Z. Wang, and E. Promislow, "Exploring local community structures in large networks," *Web Intelligence and Agent Systems: An International Journal*, vol. 6, no. 4, pp. 387–400, 2008.

[38] K. J. Lang and R. Andersen, "Finding dense and isolated submarkets in a sponsored search spending graph," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, 2007, pp. 613–622.

[39] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using pagerank vectors," in *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. IEEE, 2006, pp. 475–486.

[40] A. Clauset, "Finding local community structure in networks," *Physical review E*, vol. 72, no. 2, p. 026132, 2005.

[41] O. Amini, D. Peleg, S. Pérennes, I. Sau, and S. Saurabh, "On the approximability of some degree-constrained subgraph problems," *Discrete Applied Mathematics*, vol. 160, no. 12, pp. 1661–1679, 2012.

[42] Z. Yang, J. Yu, and M. Kitsuregawa, "Fast algorithms for top-k approximate string matching," in *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[43] I. F. Ilyas, G. Beskales, and M. A. Soliman, "A survey of top-k query processing techniques in relational database systems," *ACM Computing Surveys (CSUR)*, vol. 40, no. 4, pp. 1–58, 2008.

[44] Y. Kim and K. Shim, "Efficient top-k algorithms for approximate substring matching," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, 2013, pp. 385–396.

[45] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[46] X. Jian, X. Lian, and L. Chen, "On efficiently detecting overlapping communities over distributed dynamic graphs," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 2018, pp. 1328–1331.

[47] J. M. Kumpula, M. Kivelä, K. Kaski, and J. Saramäki, "Sequential algorithm for fast clique percolation," *Physical Review E*, vol. 78, no. 2, p. 026109, 2008.

[48] X. Jian, Y. Wang, and L. Chen, "Effective and efficient relational community detection and search in large dynamic heterogeneous information networks," *Proceedings of the VLDB Endowment*, vol. 13, no. 10, 2020.

[49] J. Li, H. Zhang, Z. Han, Y. Rong, H. Cheng, and J. Huang, "Adversarial attack on community detection by hiding individuals," in *Proceedings of The Web Conference 2020*, 2020, pp. 917–927.

[50] Z. Lai, Y. Peng, S. Yang, X. Lin, and W. Zhang, "Pefp: Efficient k-hop constrained s-t simple path enumeration on fpga," in *ICDE*. IEEE, 2021.

[51] X. Jin, Z. Yang, X. Lin, S. Yang, L. Qin, and Y. Peng, "Fast: Fpga-based subgraph matching on massive graphs," *arXiv preprint arXiv:2102.10768*, 2021.

[52] Y. Peng, X. Lin, Y. Zhang, W. Zhang, L. Qin, and J. Zhou, "Efficient hop-constrained s-t simple path enumeration," *The VLDB Journal*, pp. 1–24, 2021.

[53] M. Sozio and A. Gionis, "The community-search problem and how to plan a successful cocktail party," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2010, pp. 939–948.

[54] W. Cui, Y. Xiao, H. Wang, and W. Wang, "Local search of communities in large graphs," in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 2014, pp. 991–1002.

[55] Y. Fang, R. Cheng, X. Li, S. Luo, and J. Hu, "Effective community search over large spatial graphs," *Proceedings of the VLDB Endowment*, vol. 10, no. 6, pp. 709–720, 2017.

[56] Y. Fang, Y. Yang, W. Zhang, X. Lin, and X. Cao, "Effective and efficient community search over large heterogeneous information networks," *Proceedings of the VLDB Endowment*, vol. 13, no. 6, pp. 854–867, 2020.

[57] Q. Liu, M. Zhao, X. Huang, J. Xu, and Y. Gao, "Truss-based community search over large directed graphs," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 2183–2197.

[58] L. Chen, C. Liu, R. Zhou, J. Li, X. Yang, and B. Wang, "Maximum co-located community search in large scale social networks," *Proceedings of the VLDB Endowment*, vol. 11, no. 10, pp. 1233–1246, 2018.

[59] R.-H. Li, Q. Dai, G. Wang, Z. Ming, L. Qin, and J. X. Yu, "Improved algorithms for maximal clique search in uncertain networks," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 1178–1189.

[60] C. Zhang, W. Zhang, Y. Zhang, L. Qin, F. Zhang, and X. Lin, "Selecting the optimal groups: Efficiently computing skyline k-cliques," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 1211–1220.

[61] C. Zhang, F. Zhang, W. Zhang, B. Liu, Y. Zhang, L. Qin, and X. Lin, "Exploring finer granularity within the cores: Efficient (k, p)-core computation," in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 2020, pp. 181–192.

[62] Y. Peng, Y. Zhang, W. Zhang, X. Lin, and L. Qin, "Efficient probabilistic k-core computation on uncertain graphs," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 2018, pp. 1192–1203.

[63] T. Akiba, Y. Iwata, and Y. Yoshida, "Linear-time enumeration of maximal k-edge-connected subgraphs in large networks by random contraction," in *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 2013, pp. 909–918.

[64] L. Chang, J. X. Yu, L. Qin, X. Lin, C. Liu, and W. Liang, "Efficiently computing k-edge connected components via graph decomposition," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, 2013, pp. 205–216.

[65] J. Cheng, Y. Ke, S. Chu, and M. T. Özsu, "Efficient core decomposition in massive networks," in *2011 IEEE 27th International Conference on Data Engineering*. IEEE, 2011, pp. 51–62.

[66] W. Khaouid, M. Barsky, V. Srinivasan, and A. Thomo, "K-core decomposition of large networks on a single pc," *Proceedings of the VLDB Endowment*, vol. 9, no. 1, pp. 13–23, 2015.

[67] J. Wang and J. Cheng, "Truss decomposition in massive networks," *arXiv preprint arXiv:1205.6693*, 2012.

[68] Y. Che, Z. Lai, S. Sun, Y. Wang, and Q. Luo, "Accelerating truss decomposition on heterogeneous processors," *Proceedings of the VLDB Endowment*, vol. 13, no. 10.

[69] S. Haykin and Z. Chen, "The cocktail party problem," *Neural computation*, vol. 17, no. 9, pp. 1875–1902, 2005.

[70] J. H. McDermott, "The cocktail party problem," *Current Biology*, vol. 19, no. 22, pp. R1024–R1027, 2009.

[71] A. R. Conway, N. Cowan, and M. F. Bunting, "The cocktail party phenomenon revisited: The importance of working memory capacity," *Psychonomic bulletin & review*, vol. 8, no. 2, pp. 331–335, 2001.

[72] A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis, and S. Leonardi, "Online team formation in social networks," in *Proceedings of the 21st international conference on World Wide Web*, 2012, pp. 839–848.

	#paper	#citation	H-Index	G-Index	i-10
Michael Stonebraker	534	42173	98	200	-
Michael J. Carey	447	23637	81	148	195
David J. DeWitt	428	41844	99	203	178
H.V. Jagadish	687	36540	89	183	306
Rakesh Agrawal	542	124039	103	351	303
Michael J. Franklin	411	65161	109	254	55
Hector Garcia Molina	748	102226	149	313	437
Hamid Pirahesh	117	13114	46	114	73
Jim Gray	279	52829	83	229	167
Calvin R. Maurer Jr.	28	3671	19	28	-
Paul Suetens	401	14510	53	116	270
Max A. Viergever	589	32589	71	176	518
Derek L. G. Hill	286	14614	53	120	501
David J. Hawkes	45	2095	20	45	22
Christian S. Jensen	854	34946	93	171	364
Jennifer Widom	383	62316	109	249	208
Richard T. Snodgrass	322	14164	56	116	167
Philip A. Bernstein	347	36099	77	189	199
M. Tamer Ozsu	462	27183	64	162	183
Kyu-Young Whang	200	5176	32	69	71
Vimla L. Patel	364	16109	69	122	272
Robert A. Greenes	173	4557	30	66	-
Mario Stefanelli	147	2841	29	51	-
Edward H. Shortliffe	29	5572	20	29	210
David B. Lomet	183	4920	43	66	-
Guy M. Lohman	173	12199	59	110	117
Kenneth A. Ross	315	11266	51	103	122
Patrick Valduriez	569	19296	55	133	210
Timos K. Sellis	405	17047	55	125	197
Graeme P. Penney	106	3695	26	60	-

TABLE IV: Details of each author in the case study. The data is obtained from Aminer.

Methods \ r	5	10	15	20
naive	1	1	1	1
improved	1	1	1	1
greedy	0.9352	0.9213	0.9019	0.8736
approx	0.9999607	0.999877	0.999838	0.9998048

TABLE V: The NDCG of different algorithms on DomainPub when $k = 6, \epsilon = 0.1$.

X. Appendix

A. Additional Applications

(4) *Cocktail Party.* We aim to plan a successful cocktail party [69]–[71]. However, the size of the party is limited. Thus, we want a size-constraint dense subgraph and the influence value of the subgraph is maximum. Considering this, we could assign an influence value of each vertex in the graph, and identify a size-constraint influential community. The influence value of the community is calculated by the sum of influence values of all vertices in the community.

(5) *Team Formation [72].* Given a social network with weighted vertices, the weight indicates each vertex's capability. We are looking for a subgraph where their abilities are high and their cooperation is excellent. Additionally, the size of the subgraph is constrained. For example, a basketball team cannot exceed 15 players, whereas a soccer team cannot exceed 24 members. Furthermore, it is not uncommon for us to establish more than one team to attend the contest. Also, there should not exist overlaps between any two teams.

B. Additional Experiments

In the case study, as for the top 1 result set, the avg and sum is the same. As for min, "David J. DeWitt", "Rakesh Agrawal", and "H.V Jagadish" are in its result set, while "Jim Gray" and "Hamid Piranesh" are in the avg and sum result sets. It is observed that i-10 index of min is significantly larger than sum and avg. As for sum and avg, the top-2 results are almost the same except "Kyu-Young Whang". Compared with other results, "Kyu-Young Whang" has fewer citations, H-index, G-index, and i-10 index. Thus, it is observed that sum prefers to discover more diversified research groups, while avg could discover a communties with higher G-index.

Quality Comparison for size unconstrained sum. It is shown in Table V that NDCG of $approx$ method is nearly exact. As for the $greedy$ algorithm, it could also achieve about high NDCG. When r grows, the NDCG value decreases for all the algorithm.

Quality Comparison of $approx$. Table VI illustrates the quality of $approx$ when varying k and r. The overall trend of these two parameters is that when k or r grows, the quality of $approx$ decreases. Table VII shows that result quality when varying k and ϵ. What stands out in this table is that when ϵ grows, the NDCG of $approx$ would decrease.

As for avg, we conduct experiments on small dataset, and then $naive$, $improved$, $localsearch$ ($greedy$) are compared. **Quality Comparison of $localsearch$.** Both effectiveness and efficiency are compared in the Table VIII and IX, respectively. It is shown that $local\ search$ ($greedy$) could return feasible result.

Varying Aggregate Functions. We also compare the effectiveness and efficiency of $exact$, $improved$, $greedy$ ($local\ search$), and $random$. Tables X and Table XI illustrate the NDCG and runtime. It could be observed that our $greedy$ algorithm could return a feasible result set (at least 0.5737). As for the efficiency, the $greedy$ is almost the fastest one among all the algorithms.

A summary would be given for those parameters and some guidance would be given.

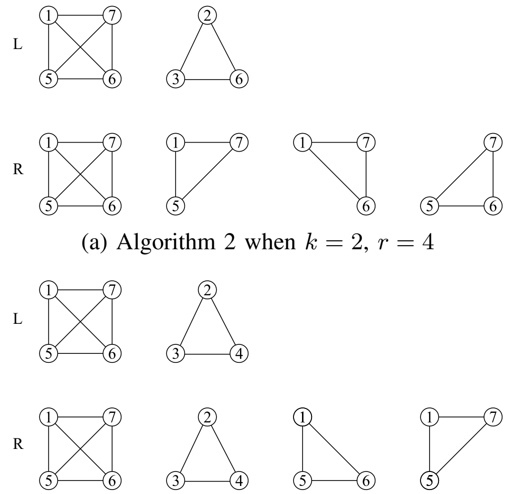
Summary of Parameters. Since there are several parameters, we discuss how to select these parameters. As for ϵ, 0.1 is feasible for most cases. When ϵ increases, the quality of result would decrease with a shorter runtime. As for k and r, $k = 6$ and $r = 20$ are suitable for most cases. The trend of

k \ r	5	10	15	20
4	1	0.9999895	0.9999886	0.99991372
6	0.9999607	0.999877	0.999838	0.9998048
8	0.999240	0.999658	0.999653	0.999403
10	0.99722	0.99835	0.99880	0.99867

TABLE VI: The NDCG of $approx$ on DomainPub when $\epsilon = 0.1$.

ϵ \ k	4	6	8	10
0.01	1	1	1	1
0.1	0.9999137	0.9998048	0.999403	0.99867
0.2	0.979240	0.96965	0.9653	0.94108
0.3	0.97722	0.96835	0.96480	0.93867
0.5	0.90722	0.89835	0.88045	0.85867
0.8	0.89722	0.86835	0.85880	0.83867

TABLE VII: The NDCG of $approx$ on DomainPub when $r = 20$ by varying k and ϵ.



(a) Algorithm 2 when $k = 2$, $r = 4$

(b) Algorithm 3 when $k = 2$, $r = 4$, and $\epsilon = 0.4$

Fig. 15: An example for how Algorithms 2 and 3 run

$\|V\|$ \ γ	2.1	2.2	2.3	2.4	2.5
50,000	0.05	0.03	0.03	0.03	0.02
100,000	0.08	0.07	0.07	0.05	0.05
150,000	0.14	0.12	0.1	0.09	0.08
200,000	0.2	0.17	0.15	0.13	0.11
250,000	0.26	0.22	0.19	0.17	0.16

TABLE XII: The runtime (second) of Algorithm 3 on Power-Law synthetic graphs when $k = 6, r = 20, \epsilon = 0.1$.

TABLE XIII: Submodular and Monotonic for different Aggregation Functions

Aggregation functions	**Submodular $f(H)$**	**Monotonic**
Minimum	✗	✓
Maximum	✓	✓
Sum	✓	✓
Sum-surplus	✓	✓
Average	✗	✗
Weight Density	✗	✓
Balanced Density	✗	✗

Algorithms \ r	5	10	15	20
naive	1	1	1	1
improved	0.8078	0.6750	0.5732	0.5081
greedy	0.8863	0.8389	0.8039	0.7782

TABLE VIII: The NDCG of avg on DomainPub when $k = 6$ and $\epsilon = 0.8$.

Algorithms \ r	5	10	15	20
exact	4.32	4.31	4.16	4.15
improved	1.93	1.94	1.92	1.95
greedy	0.64	0.61	0.6	0.58

TABLE IX: The runtime of $approx$ on DomainPub when $k = 6$.

these two parameters is similar to that of ϵ. It would be same for s, which suggested to be 20 in these datasets.

Example 3. *Figure 15 illustrates how Algorithm 2 and 3 runs and the intuition and the efficiency of Algorithm 3. Note that the aggregate function here is* sum *and the number in each vertex is the influence value.* L *is the set of the disjoint connected components after core decomposition. As for Algorithm 2, it needs to traverse all the possible candidates (the number of candidates is* 5 *in this example, but could be large in real graph) and output the exact top* 4 *result. The influence value of the first element in result set in Figure 15a equals to* $1+5+6+7 = 19$. *Thus, the top* 4 *influence values of results are* $19, 18, 14, 13$. *Nevertheless, for Algorithm 3, we only need to traverse the feasible results set. As shown in Figure 15b, the top* 1 *influence value is* 19, LB *would be* $19 \times (1 - \epsilon) = 11.4$. *Then, if any* 4 *results with influence values larger than* 11.4 *would terminate the algorithm.*

Experiments on Power-Law Graphs. *In this experiment, we evaluate the runtime by varying the* γ *and* $|V|$. *It is illustrated in Table XII that when the* γ *increases, the runtime of Algorithm would decrease. This satisfies the time complexity analysis after Lemma 1.*

functions \ methods	exact	improved	greedy	random
avg	1	0.5081	0.7782	0.15
sum	1	1	0.8736	0.21
min	1	0.575	0.83	0.11
max	1	0.7406	0.6420	0.2742
Sum-surplus	1	0.3448	0.5737	0.2006
Weight Density	1	0.4361	0.6834	0.1899
Balanced Density	1	0.2644	0.7133	0.0597

TABLE X: The NDCG of different aggregate functions on DomainPub when $k = 6, r = 20, \epsilon = 0.8$.

functions \ methods	exact	improved	greedy	random
avg	4.32	1.92	0.58	0.58
sum	0.347	0.054	0.024	0.026
min	4.26	1.9	0.58	0.58
max	4.25	1.88	0.59	0.58
Sum-surplus	4.28	1.91	0.58	0.58
Weight Density	4.26	1.88	0.58	0.58
Balanced Density	4.3	1.67	0.59	0.59

TABLE XI: The runtime of different aggregate functions on DomainPub when $k = 6, r = 20, \epsilon = 0.8$.

C. Proof for NP-hardness

Here, we present the proofs to show when the aggregation function is weight density and balanced density.

Theorem 7. *When* $f(\cdot)$ *is weight density, the top-*r k*-influential community search problem is NP-hard.*

Proof. First, we set $\beta = 1$ here to show that there does not exist any algorithm could solve the top-r k-influential community search problem in polynomial time when the aggregation function is weight density and $\beta = 1$. Given a graph $G = (V, E, w)$, we assign each vertex $v_i \in V$ with weight 0. Then, we build another graph $G' = (V', E', w')$ by adding a new vertex u that connects all vertices in V. We set the weight of the new vertex u as w_c. Suppose that there exists a polynomial-time algorithm to address the top-r k-influential community search problem. Then, we could determine whether there exists a (k-1)-clique since the influence value of top-1 k-influential community is $w_c - k$ if there exists a (k-1)-clique in graph G. Notably, adding any new vertex (or vertices) into such a clique would only increase the denominator of the influence value. However, the decision version of maximum clique search problem is NP-complete. It is a contradiction. Thus, top-r k-influential community search problem is NP-hard, when $f(\cdot)$ is weight density. □

Theorem 8. *When $f(\cdot)$ is balanced density, the top-r k-influential community search problem is NP-hard.*

Proof. Given a graph $G = (V, E, w)$, we assign each vertex $v_i \in V$ with weight 1 and $|V| = n$. Then, we build another graph $G' = (V', E', w')$ by adding a new vertex u that connects all vertices in V. We set the weight of the new vertex u as w_c. Suppose that there exists a polynomial-time algorithm to address the top-r k-influential community search problem. Then, we could determine whether there exists a ($\lceil \frac{n-w_c}{2} \rceil$-1)-clique since the influence value of top-1 k-influential community is $(w_c + k)/(w_c + 2k - n)$, where $k = \lceil \frac{n-w_c}{2} \rceil$, if there exists a ($\lceil \frac{n-w_c}{2} \rceil$-1)-clique in graph G. Notably, adding any new vertex (or vertices) into such a clique would only increase the denominator of the influence value. However, the decision version of maximum clique search problem is NP-complete. It is a contradiction. Thus, top-r k-influential community search problem is NP-hard, when $f(\cdot)$ is balanced density. □