



Critical Nodes Identification in Large Networks: The Inclined and Detached Models

Renjie Sun¹ · Chen Chen² · Xijuan Liu² · Shuangyan Xu² · Xiaoyang Wang² · Xuemin Lin³

Received: 10 January 2022 / Revised: 10 January 2022 / Accepted: 18 March 2022 /

Published online: 25 April 2022

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract

In social networks, the departure of some users can lead to the dropout of others from the community in cascade. Therefore, the engagement of critical users can significantly influence the stability of a network. In the literature, the anchored/collapsed k -core problem is proposed, which aims to enlarge/collapse the community by anchoring/deleting certain nodes. While, in real social networks, nodes are usually associated with different preferences, such as close or conflict interest. Intuitively, a community will be more stable if more nodes share close interest and fewer of them carry conflict interest. However, most existing researches simply treat all users equally, and the inclination property is neglected. To fill the gap, in this paper, we propose two novel problems, named inclined anchored k -core (IAK) problem and minimum detached k -core (MDK) problem, to better characterize the real scenarios. We prove that both problems are NP-hard. To facilitate the computation, novel search strategies are proposed. Comprehensive experiments are conducted on 9 networks to demonstrate the effectiveness and efficiency of the proposed techniques.

Keywords Inclined anchored k -core · Minimum detached k -core · NP-hard

1 Introduction

In recent years, modeling user engagement for user behavior has attracted great interest from researchers [2, 6, 7, 10–12, 28]. In social network analysis, the k -core model is widely used in the study of network stability or engagement, e.g., [2, 7, 28]. Given a graph G , k -core is the maximal subgraph, in which each node has at least k neighbors. The size of k -core plays an important role in graph analysis. To enlarge (resp. collapse) the size of k -core, the anchored (collapsed) k -core problem are proposed in the literature [2, 29]. By giving users some incentives, the anchored k -core problem aims to anchor b nodes,

This article belongs to the Topical Collection: *Special Issue on Web Information Systems Engineering 2021*

Guest Editors: Hua Wang, Wenjie Zhang, Lei Zou, and Zakaria Maamar

✉ Chen Chen
chenc@zjgsu.edu.cn

Extended author information available on the last page of the article

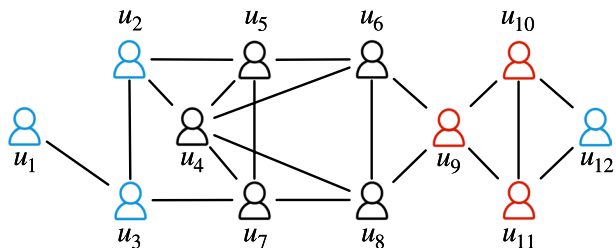
i.e., stay in k -core regardless of the neighbor size constraint, in order to maximize the corresponding k -core size. The selected nodes are called anchored nodes. On the contrary, the collapsed k -core problem attempt to delete certain nodes in order to minimize the corresponding k -core. In real applications, users in a network are usually associated with different properties, such as close or conflict interest, towards the community. However, the previous models treat each node equally and may fail to characterize the propensity and stability of the network.

Intuitively, a community is more stable if more nodes inside share close interest and fewer nodes hold conflict interest [3]. It will also assist more new users to join the community if the community is strongly inclined. Motivated by this, in this paper, we propose two new models, i.e., inclined anchored k -core(IAK) and minimum detached k -core(MDK), to fill the gap. For the IAK problem, we consider the nodes outside the k -core are marked with labels, i.e., close, conflict or unlabeled. When anchoring nodes, both nodes with close and conflict interest may join the k -core community. In the IAK problem, we aim to find b anchor nodes, such that more close nodes but fewer conflict nodes will involve in the anchored k -core. For MDK problem, we consider a set \mathcal{E} of nodes in the k -core are attached with conflict interest. To maintain the harmony in the community, we aim to collapse or detach the minimum number of nodes, such that all the nodes in \mathcal{E} are removed from the k -core. As observed, we not only consider the size of resulting k -core, but also pore over the inclination of the returned community. Through this way, we can better portray the resilience of networks. Following are two motivating examples.

Example 1 As shown in Figure 1, there is a social network with 12 users and their corresponding connections. Suppose $k = 3$, the close node set $F = \{u_1, u_2, u_3, u_{12}\}$ (blue nodes), the conflict node set $\mathcal{E} = \{u_9, u_{10}, u_{11}\}$ (red nodes), and the others are marked as unlabeled. The willingness of a user to engage in the group depends on the number of his neighbors. If a user leaves the group, the willingness of his neighbors will decrease, which may lead to a cascade of others to drop out the community. Given $k = 3$, only nodes u_4, u_5, \dots, u_8 belong to the 3-core. To enhance the inclination and stability of the group, we can anchor the critical nodes by giving them incentives. If we anchor u_1 , there will be three close followers $\{u_1, u_2, u_3\}$ (i.e., join the anchored k -core) and no conflict followers. When u_{12} is anchored, there will be four followers $\{u_9, u_{10}, u_{11}, u_{12}\}$, but three of them are from the conflict group. Thus, for $b = 1$, we can select u_1 as the optimal node to anchor. It can also be observed, simply enlarging the k -core size will not guarantee the increase of nodes with close interest.

Example 2 Reconsider the graph in Figure 1. Suppose the conflict node set $\mathcal{E} = \{u_9, u_{10}, u_{11}\}$ and $k = 2$. Then, all nodes in the graph except u_1 belong to the 2-core. To make the group more harmonious, i.e., not contain any conflict nodes, we need to detach some nodes from the group. Considering the cost of detaching, we need to choose as few detached nodes as possible. Thus, in the example, $\{u_9, u_{12}\}$ is the best detached node set. After deleting $\{u_9, u_{12}\}$, all the nodes in the conflict node set will be removed from the 2-core.

Figure 1 Motivation example



Challenges and Contributions In this paper, we propose and investigate the inclined anchored k -core and minimum detached k -core problems. We prove that both problems are NP-hard, which implies that it is non-trivial to solve them within polynomial time. Thus, following the routine in previous studies, in this paper, we aim to design heuristic strategies that can return competitive results efficiently. In addition, due to the different properties of the objective functions, lots of pruning rules in previous studies no longer hold for our problems. Secondly, in real-world social networks, the search space is usually quite large, which is time-consuming to conduct the exploration. To enhance the processing, different search approaches and pruning techniques are considered. The main contributions of the paper are summarized as follows.

- To the best of our knowledge, we are the first to investigate the inclined anchored k -core and minimum detached k -core problems. We formally define the problems and prove the hardness of the problems.
- For the inclined anchored k -core problem, a layer-based method is adopted to accelerate the exploration, and upper bound based techniques are integrated to further speedup the procedure.
- For the minimum detached k -core problem, two heuristic strategies, i.e., conflict remove heuristic and degree decline heuristic, are proposed by leveraging different properties of the model.
- We conduct extensive experiments over 9 datasets to evaluate the efficiency and effectiveness of proposed methods.

Roadmap The rest of this paper is organized as follows. In Sect. 2, we formally define the problems studied and show their properties. In Sect. 3, we present the solutions for the IAK problem. In Sect. 4, novel heuristic strategies are proposed for MDK problem. We report the experiment results in Sect. 5. We introduce the related work in Section 6 and conclude the paper in Sect. 7.

2 Preliminaries

In this section, we first formally define the inclined anchored k -core and minimum detached k -core problems. Then, we present the analysis about the properties and hardness of the problems. Table 1 summarizes the notations frequently used throughout the paper.

2.1 Problem definition

We consider an unweighted and undirected graph $G = (V, E)$, where V and E represent the sets of nodes and edges, respectively. We denote $n = |V|$ and $m = |E|$. Given a subgraph $S \subseteq G$, $N(u, S)$ is the set of adjacent nodes of u in S . $deg(u, S)$ is the degree of u in S , i.e., $deg(u, S) = |N(u, S)|$. To measure the cohesiveness of subgraph, we employ the concept of k -core in this paper.

Definition 1 Given a graph G , a subgraph S is the k -core of G , denoted by $C_k(G)$, if 1) $deg(u, S) \geq k$ for each node $u \in S$; 2) S is maximal, i.e., any supergraph $S' \supset S$ is not a k -core.

Table 1 Summary of notations

Notation	Definition
G	an unweighted and undirected graph
$V; E$	the node set of G ; the edge set of G
n	the number of nodes in G
m	the number of edges in G
S	a subgraph of G
$N(u, S)$	the set of adjacent nodes of u in S
$deg(u, S)$	the number of adjacent nodes of u in S
$core(u)$	the coreness of u
$C_k(G); \mathcal{P}_k(G)$	the k -core of G ; the k -peel of G
\mathcal{A}	a set of anchored nodes
\mathcal{D}	a set of detached nodes
$\mathcal{F}; \mathcal{E}$	the close node set; the conflict node set
$\mathcal{F}(u, G)$	the followers of the anchored node u
$\mathcal{F}^+(\mathcal{A}, G); \mathcal{F}^-(\mathcal{A}, G)$	the close (resp. conflict) followers of \mathcal{A}
$Score(\mathcal{A}, G)$	the difference of $\mathcal{F}^+(\mathcal{A}, G)$ and $\mathcal{F}^-(\mathcal{A}, G)$

Definition 2 Given a graph G , the coreness of a node $u \in G$, denoted by $core(u)$, is the largest k such that u belongs to k -core, i.e., $core(u) = \max\{k | u \in C_k(G)\}$.

The k -core of a graph G can be obtained by recursively removing the node whose degree is less than k with a time complexity of $\mathcal{O}(m)$ [1, 22]. The pseudo code is shown in Algorithm 1.

Algorithm 1: Compute k -core

Input : G : a graph, k : degree constraint
Output : $C_k(G)$: the k -core of G

```

1 while  $\exists u \in V$  with  $deg(u, G) < k$  do
2   for each  $v \in N(u, G)$  do
3      $deg(v, G) \leftarrow deg(v, G) - 1$ ;
4     remove  $u$  and its adjacent edges from  $G$ ;
5 return  $G$ 

```

Definition 3 Given a graph G , the k -peel of G , denoted by $\mathcal{P}_k(G)$, is the set of nodes that have coreness equal to k , i.e., $\mathcal{P}_k(G) = C_k(G) \setminus C_{k+1}(G)$.

The k -core model is widely used to measure the properties of the graph. In order to enlarge the k -core, we can offer some incentives to a set \mathcal{A} of nodes, named the anchored nodes, to make them stay in the k -core community. That is, once a node is anchored, it is always reserved in the k -core, i.e., with infinite large degree. Through anchoring these nodes, some other nodes will join the k -core in cascade and finally the k -core is enlarged.

Definition 4 Given a graph G and an anchored node set $\mathcal{A} \subseteq V$, the anchored k -core, denoted by $C_k(G \oplus \mathcal{A})$, is the corresponding k -core of G with nodes in \mathcal{A} anchored.

We use $\mathcal{F}(u, G)$ to denote the set of nodes that join the k -core when u is anchored, i.e., $\mathcal{F}(u, G) = C_k(G \oplus u) \setminus C_k(G)$, and call $\mathcal{F}(u, G)$ as the followers of u . As discussed, a community may have its own preferences, such as nodes with close or conflict interests. When selecting anchored nodes, we would like to preserve more users with close interests and less users with conflict interests. Given the set F (resp. \mathcal{E}) of nodes with **close** (resp. **conflict**) interests, we use $\mathcal{F}^+(\mathcal{A}, G)$ (resp. $\mathcal{F}^-(\mathcal{A}, G)$) to denote the close (resp. conflict) followers of \mathcal{A} , i.e., $\mathcal{F}(\mathcal{A}, G) \cap F$ (resp. $\mathcal{F}(\mathcal{A}, G) \cap \mathcal{E}$). To enlarge a community through anchoring nodes, intuitively, we want to involve more users with close interests and fewer ones with conflict interests. Then, we define the inclined score as follows.

Definition 5 [inclined score] Given a graph G , an anchored node set \mathcal{A} , and the close and conflict node sets F and \mathcal{E} , the inclined score of \mathcal{A} , denoted as $\text{Score}(\mathcal{A}, G)$, is the difference between the number of close followers and that of conflict followers, i.e., $\text{Score}(\mathcal{A}, G) = \mathcal{F}^+(\mathcal{A}, G) - \mathcal{F}^-(\mathcal{A}, G)$.

We propose the concept of inclined score to better judge the effectiveness of anchored nodes by considering the inclination. When the context is clear, we omit the second parameter, i.e., G , from the notations.

Problem 1 (IAK problem) Given a graph G , the close and conflict node sets F and \mathcal{E} , the degree constraint k and a budget b , the inclined anchored k -core (IAK) problem aims to anchor a set of b nodes \mathcal{A}^* with the largest inclined score, i.e.,

$$\mathcal{A}^* = \arg \max_{\mathcal{A} \subseteq G \wedge |\mathcal{A}|=b} \text{Score}(\mathcal{A})$$

Example 3 Reconsider the graph in Figure 1. Assume $k = 3$ and $b = 1$. The close and conflict node sets are $F = \{u_1, u_2, u_3, u_{12}\}$ and $\mathcal{E} = \{u_9, u_{10}, u_{11}\}$, respectively. By anchoring the node u_1 , we can obtain the anchored k -core $\{u_1, u_2, \dots, u_8\}$ with inclined score 3. The inclined score is -2, if we anchor u_{12} .

Contrary to the anchor k -core problem, the dropout of certain nodes could collapse the k -core community, i.e., the collapsed k -core problem. As discussed, sometimes we may enforce the leaving of certain nodes to detach the conflict users from the community. Motivated by this, we introduce the minimum detached k -core problem as follows.

Problem 2 (MDK problem) Given a graph G , the conflict node set \mathcal{E} and the degree constraint k , the minimum detached k -core (MDK) problem aims to find the smallest node set of $\mathcal{D}^* \subseteq G$, such that after the deletion of \mathcal{D}^* , the k -core of G contains no nodes in \mathcal{E} . That is, $C_k(G \setminus \mathcal{D}^*)$ does not contain any nodes in \mathcal{E} , i.e.,

$$\mathcal{D}^* = \arg \min_{\mathcal{D} \subseteq G \wedge C_k(G \setminus \mathcal{D}) \cap \mathcal{E} = \emptyset} |\mathcal{D}|$$

Example 4 Reconsider the graph in Figure 1. Suppose $k = 2$ and the conflict node set is $\mathcal{E} = \{u_9, u_{10}, u_{11}\}$. The detached node set $\mathcal{D} = \{u_9, u_{12}\}$ is optimal result for MDK problem with size of 2, and $C_k(G \setminus \mathcal{D})$ does not contain any node in \mathcal{E} .

2.2 Problem properties

In this section, theoretical analysis is conducted about the hardness and properties of the studied problems. According to Theorems 1 and 2, the inclined anchored k -core problem is NP-hard, and the objective function is non-monotonic and non-submodular. Then, we prove the minimum detached k -core problem is also NP-hard in Theorem 3.

Theorem 1 *The IAK problem is NP-hard for $k \geq 3$.*

Proof When there are only close nodes in the graph, the inclined anchored k -core problem can be reduced to the traditional anchored k -core problem, which is NP-hard for $k \geq 3$ [2]. Hence, the inclined anchored k -core problem studied in this paper is also NP-hard.

Theorem 2 *The objective function of IAK problem is non-monotonic and non-submodular.*

Proof Non-monotonic. We prove $Score(X)$ is non-monotonic by constructing a counter example. Reconsider the graph in Figure 1. For $k = 3$, $\mathcal{E} = \{u_9, u_{10}, u_{11}\}$ and $F = \{u_1, u_2, u_3, u_{12}\}$, suppose $\mathcal{A} = \{u_3\}$. We have $Score(\mathcal{A}) = 2$. By adding u_{12} to \mathcal{A} , the inclined score becomes 0. While, the score becomes 3, if we add u_1 to \mathcal{A} . Thus, the objective function is non-monotonic.

Non-submodular. Given two sets A and B , we say the function $Score(X)$ is submodular if $Score(A) + Score(B) \geq Score(A \cup B) + Score(A \cap B)$. We also show the inequality does not hold by constructing a counter example. In Figure 1, for $k = 3$, $\mathcal{E} = \{u_9, u_{10}, u_{11}\}$ and $F = \{u_1, u_2, u_3, u_{12}\}$, suppose $A = \{u_{10}\}$ and $B = \{u_{12}\}$. We have $Score(A) = -2$, $Score(B) = -2$, $Score(A \cap B) = 0$ and $Score(A \cup B) = -2$. Hence, the inequation does not hold. Therefore, the theorem is correct.

Theorem 3 *The MDK problem is NP-hard for $k \geq 2$.*

Proof We prove this theorem by reducing set cover problem [15] to MDK problem. Given an element set $\mathcal{Q} = \{e_1, e_2, \dots, e_m\}$ with m elements and a collection $\mathcal{T} = \{T_1, T_2, \dots, T_n\}$ of n sets whose union equals \mathcal{Q} . The set cover problem aims to identify the smallest sub-collection $R \subseteq \mathcal{T}$ whose union equals \mathcal{Q} . Then we construct a graph G such that we can find a minimum detached node set \mathcal{D} for MDK problem if and only if the set cover problem is answered. Following is the construction procedure.

First, we create some nodes. For each element e_i in \mathcal{Q} , we generate a node u_i and let conflict set $\mathcal{E} = \{u_i | 1 \leq i \leq m\}$. For each set $T_j \in \mathcal{T}$, we generate a node v_j . Let $V = \{v_j | 1 \leq j \leq n\}$. We add a k -core M with at least $(m + n) \times k$ nodes. Then, we build some edges between these nodes. If an element $e_i \in T_j$, we add an edge between u_i and v_j . Let t be the max $deg(u_i, V)$ for any $u_i \in \mathcal{E}$. For each node v_j in V , let it be connected to k nodes in M . For each node u_i in \mathcal{E} , let it connected to $k - deg(u_i, V)$ nodes in M . Consequently, the construction is completed. Figure 2 is a construction example with $m = 6$, $n = 4$ and $k = 3$.

With the construction, we can ensure that *i*) M is always a k -core, no matter how many nodes in V and \mathcal{E} are deleted; *ii*) v_j always in k -core, no matter how many nodes in $V \setminus \{v_j\}$ and \mathcal{E} are deleted; *iii*) once v_j is deleted, all its neighbors u_i in \mathcal{E} will not belong to k -core.

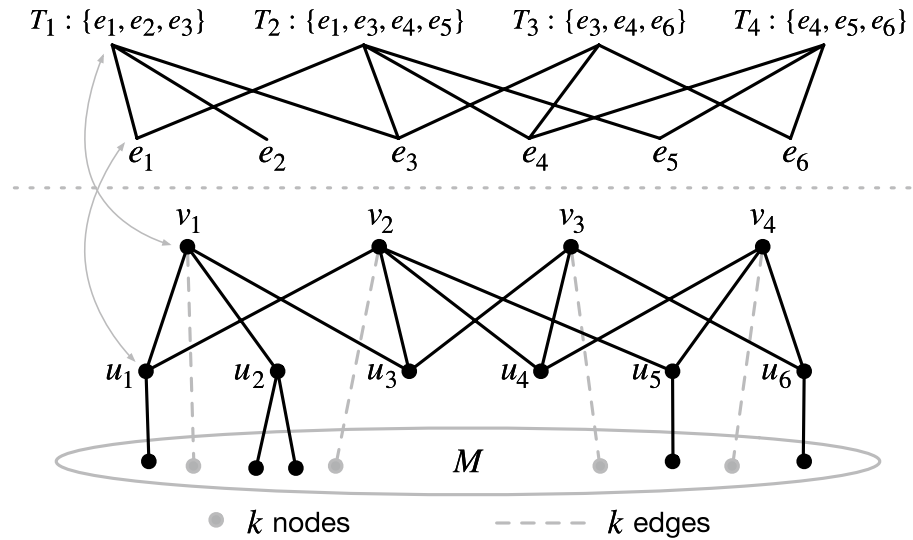


Figure 2 Construction example for NP-hard proof, where $m = 6, n = 4$ and $k = 3$

By doing this, for any $k \geq t$, the sub-set \mathcal{D} of V corresponding to the smallest sub-collection $R \subseteq \mathcal{T}$ whose union equals \mathcal{Q} of set cover problem is the optimal solution of MDK problem. Hence, this theorem is correct.

3 Solution for IAK problem

In this section, a baseline search framework is firstly proposed. Then, optimized solutions are developed to accelerate the computation.

3.1 Baseline algorithm

For the inclined anchored k -core problem, a native method is to enumerate all possible node sets \mathcal{A} with size b and compute the corresponding inclined score. Then, we just return the result with the largest score. However, the exact algorithm is time-consuming because of the large number of combinations. Due to the NP-hardness of the inclined anchored k -core problem, we resort to the greedy heuristic as the traditional anchored k -core solution does [28]. The details are shown in Algorithm 2. It iteratively finds the node with the largest score in current k -core. It is easy to verify that we only need to consider the nodes not in the k -core as candidates.

3.2 Follower computation

Because of the non-monotonic property, the pruning techniques developed for the traditional anchored k -core problem are no longer held for our inclined case. For our problem, an essential task is to compute the followers of an anchored node.

Algorithm 2: Baseline Algorithm

Input : G : a graph, k : degree constraint, b : the budget
Output : \mathcal{A} : the set of anchored nodes

- 1 $\mathcal{A} \leftarrow \emptyset$;
- 2 **for** i from 1 to b **do**
- 3 **for** each $u \in G \setminus \{\mathcal{A} \cup C_k(G)\}$ **do**
- 4 compute $Score(u, G)$;
- 5 $u^* \leftarrow$ the node with the maximum score;
- 6 $\mathcal{A} \leftarrow \mathcal{A} \cup \{u^*\}$;
- 7 **return** \mathcal{A}

Algorithm 3: Radiate Search

Input : x : the candidate anchor; P : layer structure
Output : \mathcal{F} : the followers of x

- 1 the status of x is set received;
- 2 **for** each node v in $V(G) \setminus \{x\}$ **do**
- 3 $q \leftarrow$ the number of neighbors of v in $C_k(G)$;
- 4 $dis(v) = k - q$; v is set listening;
- 5 $Q \leftarrow \emptyset$;
- 6 **for** each $w \in N(x) \cup P$ and $p(w) > p(x)$ **do**
- 7 $Q.push(w)$;
- 8 $dis(w) = dis(w) - 1$; $dis(x) = dis(x) - 1$;
- 9 **if** $dis(w) \leq 0$ **then** w is set received;
- 10 **while** $Q \neq \emptyset$ **do**
- 11 $u \leftarrow Q.pop()$;
- 12 **if** $dis(u) > a(u) + r(u) + l(u)$ **then**
- 13 u is set closed; SHRINK(u);
- 14 **else**
- 15 u is set activated;
- 16 **for** each listening node z in $N(u)$ **do**
- 17 **if** z is not in $C_k(G)$ and $p(z) > p(u)$ **then**
- 18 $Q.push(z)$;
- 19 $dis(u) = dis(u) - 1$;
- 20 $dis(z) = dis(z) - 1$;
- 21 **if** $dis(z) \leq 0$ **then** z is set received;
- 22 **return** received nodes in P

Algorithm 4: Shrink(u)

Input : u : the node for degree check

- 1 **for** each activated neighbor v of u **do**
- 2 $dis(v) = dis(v) + 1$;
- 3 **if** $dis(v) > a(v) + r(v) + l(v)$ **then**
- 4 v is set closed; SHRINK(v);
- 5 **for** each received neighbor v of u **do**
- 6 $dis(v) = dis(v) + 1$;
- 7 **if** $dis(v) > 0$ **then** v is set activated;
- 8 **if** $dis(v) > a(v) + r(v) + l(v)$ **then**
- 9 v is set closed; SHRINK(v);

Lemma 1 *By anchoring a node, all of its followers are from \mathcal{P}_{k-1} .*

Proof By anchoring a node, the coreness of a node increases at most 1 [28]. Therefore, if one node u is not in $(k - 1)$ -core, then anchoring a node x can increase the coreness of u to $k - 1$ at most, and u cannot appear in the k -core. Thus, it cannot be the follower of x .

Peel layer Based on the definition of $(k - 1)$ -peel, we can divide the nodes in \mathcal{P}_{k-1} into different layers. Motivated by the onion layer structure [28], we recursively remove the node with degree less than k and organize the nodes in \mathcal{P}_{k-1} in a peel layer structure P . Thus, the set of nodes which are deleted in the i -th batch belong to the i -layer, denoted by P_i . Specifically, $P_1 = \{u \mid \text{deg}(u, C_{k-1}(G)) < k \wedge u \in C_{k-1}(G)\}$. Due to the deletion of P_1 , we can get P_2 in the same manner. Similarly, we recursively get all P_i . So, the peel layer structure $P = \bigcup_{i=1}^t P_i$, where t represents the recursion times. We use $p(u)$ to denote the layer index of a node u , i.e., $p(u) = i$ when $u \in P_i$.

According to the peel layer structure P and Lemma 1, we have that the valid candidate anchored nodes with at least one follower must belong to P , which can significantly reduce the size of candidates. We propose an effective candidate pruning technique based on peel layer structure based on Observation 1. To better explain the pruning rule, we first present the definition of ladder path.

Definition 6 Given an anchored node x , there is a ladder path from x to u , denoted by $x \rightsquigarrow u$, where (i) all nodes are from P ; (ii) $p(v) < p(w)$ for every two consecutive nodes v and w along this path.

Observation 1 Given a graph G , if a candidate anchor x has at least one follower u , we have that there exists a ladder path $x \rightsquigarrow u$.

Clearly, if there is no one ladder path $x \rightsquigarrow u$ for a node x , it cannot be considered as a candidate anchor. Then, we compute the followers for each candidate anchor based on the radiate search after candidate reduction. We give the definition involved before presenting the details.

Definition 7 [distance] Give a graph G and a node u , the distance of u , denoted as $\text{dis}(u)$, is the difference between degree constraint k and the number of neighbors of u in k -core, i.e., $\text{dis}(u) = k - \text{deg}(u, C_k(G))$. The distance of u represents that u still need $\text{dis}(u)$ nodes to satisfy the requirement of degree not less than k .

Radiate Search We find the followers of anchors by radiate search, which is similar to breadth-first search. During the search procedure, each node has four states. When we start to anchor a node u , then other nodes will be **listening** $l(v)$. Due to the degree of anchored node u is infinity, which satisfies the degree constraint, i.e. $\text{dis}(u) \leq 0$, u is **received** $r(u)$. A node is **activated** $a(v)$ when its neighbor is received or activated. In addition, if $p(v) \geq p(u)$, then the node v cannot be activated based on the layer structure P . After anchoring a node u , the distance of each neighbor v of u subtract 1, and we check whether the distance of v is 0. If the distance is 0, then v is received, so we continue to check the neighbors of v . Otherwise, we need to check that whether the neighbors of v can participate in the k -core to decrease $\text{dis}(v)$. Note that, if $\text{dis}(v) > a(v) + r(v) + l(v)$, then the distance of v cannot be 0, thus, v is **closed** $c(v)$. The details are described in Algorithm 3.

At first, we set the status of anchored node x as received (Line 1). Then, we process all other nodes in G (Lines 2-4). For each node, we compute the degree of v in $C_k(G)$ (Line 3) and obtain the distance of v . Then, we set the status of v as listening (Line 4). After that, we set queue Q as empty in Line 5. In Lines 6-9, for each node w who is the neighbor of x or belongs to layer structure P with $p(w) > p(x)$, we push it into the queue Q (Line 7). Then, we update the distance for both w and x by subtracting 1 (Line 8). If the distance of w is no larger than 0, we set the status of it as received (Line 9). In Lines 10-21, we process all nodes in Q until it becomes empty. In Line 11, we pop a node u from Q . If the distance of u is larger than $a(u) + r(u) + l(u)$ which means the node u cannot be the follower, we set the status of u as closed, and use the SHRINK of Algorithm 4 to update its neighbors' distance and status (Lines 12-13). Otherwise, we set the status of u as activated (Line 15) and process each listening node z who is the neighbors of u (Lines 16-21). If node z is not in the k -core and $p(z) > p(u)$, we push z into Q and update the distance u and z by subtracting 1. Then, we judge the distance of z . If $dis(z)$ is no larger than 0, we set the status of z as received. Finally, we return all received nodes in P until Q is empty.

Example 5 As shown in Figure 3, given the part of a graph G and the distance $d(u)$ for each u , when we anchor the node u_3 , then u_3 is received and its neighbors are activated and other nodes are listening. The distance of u_2 subtract 1, i.e., $d(u_2) = 4$ and the distance of neighbors $\{u_5, u_7, u_4, u_1\}$ of u_2 subtract 1. We have that $d(u_5) = 0$, $d(u_2) = 0$. Thus, the node u_2 and u_5 are received. Similarly, activating u_1 can activate its neighbors and make their distance subtract 1. Finally, we find that all nodes are received except u_8 .

Algorithm 5: KSM Algorithm

Input : G : a graph, k : degree constraint, b : the budget
Output : \mathcal{A} : the set of anchored nodes

- 1 $\mathcal{A} \leftarrow \emptyset$;
- 2 Compute upper bounds of inclined score;
- 3 **for** i from 1 to b **do**
- 4 $\alpha \leftarrow -\infty$;
- 5 **for each** u in P with at least one ladder path **do**
- 6 **if** $u \notin \mathcal{A}$ and $f^+(u) > \alpha$ **then**
- 7 compute $Score(u, G)$ based on Radiate Search with upper bound;
- 8 update α if needed;
- 9 $u^* \leftarrow$ the nodes with maximum score;
- 10 $\mathcal{A} \leftarrow \mathcal{A} \cup \{u^*\}$;
- 11 Refine upper bounds;
- 12 **return** \mathcal{A}

3.3 Search algorithm

Let $CC(u)$ be the connected component of u . The number of close nodes in $CC(u)$ is denoted by $f^+(u)$. Then, we can use $f^+(u)$ to serve as an upper bound to filter the candidate. It is easy to verify the correctness of Lemma 2.

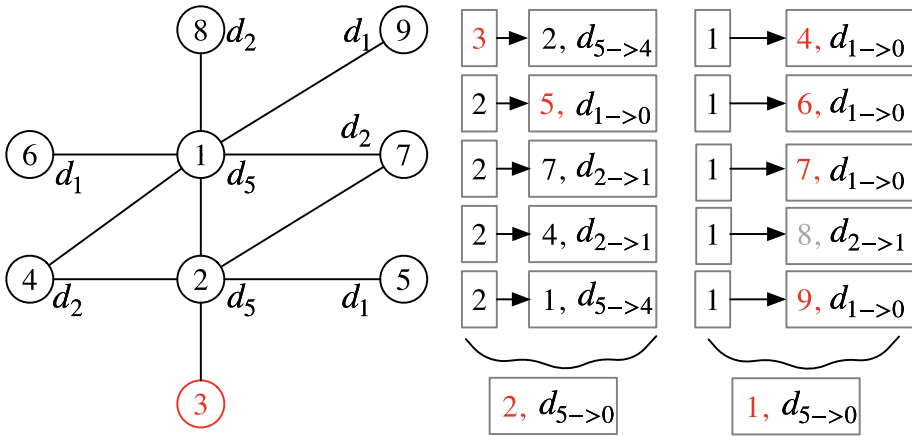


Figure 3 Example of radiate search

Lemma 2 Let α denote the largest marginal score currently after anchoring a candidate node. If $f^+(u) < \alpha$, we can remove u from the candidate.

By further extending the result in Lemma 2, we can maintain a tighter upper bound when conducting the radiate search. That is, when exploring the layer structure, we can decrease the upper bound value $f^+(u)$ by if we meet a node in the conflict set. Then, we can terminate the computation when the updated upper bound violates Lemma 2. By integrating all optimization techniques, we present the KSM algorithm as shown in Algorithm 5. We skip a node u , if $f^+(u)$ is no large than α . Otherwise, we conduct a radiate search that integrates the extended Lemma 2. At the end of current iteration, we have the best anchor u^* with maximum score and merge it into \mathcal{A} . Lastly, Algorithm 5 returns the set \mathcal{A} of b anchored nodes after b iterations.

4 Solution for MDK problem

In this section, we first present an exact algorithm by enumerating all the combinations. Then, we propose two heuristic algorithms based on the properties of the problem.

4.1 Exact Algorithm

Recall that the MDK problem is to find the smallest node set \mathcal{D} from G so that $C_k(G \setminus \mathcal{D})$ does not contain any nodes in the conflict node set \mathcal{E} .

Observation 2 The deletion of node in $G \setminus C_k(G)$ will not influence the structure of k -core.

Observation 3 Given a graph G and the conflict set \mathcal{E} , for the MDK problem, the size of the optimal solution \mathcal{D} is no larger than that of \mathcal{E} .

The observation is easy to verify, since 1) the node in $G \setminus C_k(G)$ will be deleted when computing the k -core, and 2) $C_k(G \setminus \mathcal{E})$ must not contain \mathcal{E} . Therefore, to find the optimal

solution of MDK, we can first compute the k -core of G , and then enumerate all subsets $\mathcal{D}' \subseteq V(C_k(G))$ with size from 1 to $|\mathcal{E}|$. Then, we select the first encountered node set \mathcal{D}' such that $C_k(G \setminus \mathcal{D}')$ does not contain any nodes in \mathcal{E} . In the worst case, the time complexity of the algorithm is $\mathcal{O}(2^{|\mathcal{E}|} |E(C_k(G))|)$. The above solution is too cost to be adopted in real applications, especially when $|\mathcal{E}|$ is large. In the next sections, we turn to heuristic solutions, which can return a competitive result efficiently.

Algorithm 6: CRH Algorithm

```

Input :  $G$ : a graph,  $k$ : degree constraint,  $\mathcal{E}$ : conflict node set
Output :  $\mathcal{D}$ : the set of detached nodes
1  $G' \leftarrow C_k(G)$ ;
2  $\mathcal{D} \leftarrow \emptyset$ ;
3 while  $\mathcal{E} \neq \emptyset$  do
4    $\mathcal{E}_k \leftarrow \{v | v \in \mathcal{E} \wedge \text{deg}(v, G') = k\}$ ;
5    $N_k \leftarrow \{u | v \in \mathcal{E}_k \wedge u \in N(v, G')\}$ ;
6    $u \leftarrow \text{none}$ ;  $\alpha \leftarrow -\infty$ ;
7   if  $\mathcal{E}_k \neq \emptyset$  then
8     for each  $v \in N_k$  do
9       if  $v \in \mathcal{E}$  then
10        if  $|N(v, G') \cap \mathcal{E}_k| + 1 > \alpha$  then
11           $\alpha \leftarrow |N(v, G') \cap \mathcal{E}_k| + 1$ ;
12           $u \leftarrow v$ ;
13        else
14          if  $|N(v, G') \cap \mathcal{E}_k| > \alpha$  then
15             $\alpha \leftarrow |N(v, G') \cap \mathcal{E}_k|$ ;
16             $u \leftarrow v$ ;
17      else
18         $u \leftarrow$  the node in  $\mathcal{E}$  with max coreness;
19       $\mathcal{D} \leftarrow \mathcal{D} \cup \{u\}$ ;
20       $G' \leftarrow C_k(G' \setminus \mathcal{D})$ ;
21      update  $\mathcal{E}$ ;
22 return  $\mathcal{D}$ 

```

4.2 Conflict removing heuristic

For any node $v \in V(C_k(G))$, if we want v to not be contained in $C_k(G)$, there are two approaches, 1) one is to delete v directly, and 2) the other is to delete $\text{deg}(v, C_k(G)) - k + 1$ v 's neighbors in $C_k(G)$. Considering that we aim to find the smallest node set \mathcal{D} , the latter method usually requires more nodes than the former method, unless $\text{deg}(v, C_k(G)) = k$. Before introducing the detailed heuristic method, we first show a property of k -core in Lemma 3.

Lemma 3 *Let S be a subset of k -core but not in $(k + 1)$ -core of G . Then, there must exist a node u in S such that $S \setminus u$ does not satisfy the degree constraint of k -core.*

Proof Let v be a node in S with the minimum degree. Then, $\text{deg}(v, S)$ must equal k , otherwise S should also be inside $(k + 1)$ -core. Suppose u is a neighbor of v in S . After u is removed, the degree of v in S will be $k - 1$. Thus, $S \setminus u$ cannot be a k -core, and the lemma holds.

Since $\mathcal{D} = \mathcal{E}$ is a solution for MDK problem, although not necessarily the optimal one, in this heuristic method, we try to optimize this solution by shrinking its

cardinality. According to Lemma 3, for a node $v \in \mathcal{E} \cap \mathcal{P}_k$, i.e., $deg(v, C_k(G)) = k$, if one of its neighbor u in $C_k(G)$ is deleted, v cannot stay in $C_k(G)$ due to the degree constraint. Moreover, all neighbors of u in $\mathcal{E} \cap \mathcal{P}_k$ cannot exist in $C_k(G)$ either. Therefore, $\mathcal{D}' = (\mathcal{E} \setminus (\mathcal{E} \cap \mathcal{P}_k \cap N(u, G))) \cup \{u\}$ is a superior solution compared with $\mathcal{D} = \mathcal{E}$. Based on this idea, we propose our conflict removing heuristic (CRH) algorithm, which tries to reduce the result size iteratively. The details of this algorithm is shown in Algorithm 6.

In Algorithm 6, we first compute the k -core of G (line 1). Then, we iteratively select nodes to be removed from $C_k(G)$ until $C_k(G)$ does not contain any node in \mathcal{E} (lines 3-21). Let \mathcal{E}_k be the nodes in \mathcal{E} with degree equal k (line 4) in the k -core, and N_k be the neighbors of each node in \mathcal{E}_k (line 5). If there exist nodes in \mathcal{E}_k , we delete the node from N_k that has the largest number of neighbors in \mathcal{E}_k . Otherwise, we delete the node from \mathcal{E} that has the largest coreness (lines 7-18). Note that, the nodes in N_k may also belong to \mathcal{E} , so they need to be handled differently (lines 9-16). After each iteration, we update k -core and \mathcal{E} (lines 20-21). Finally, we returned the detached node set \mathcal{D} in line 22.

4.3 Degree decline heuristic

The CRH algorithm is based on the idea that to make a node $v \in V(C_k(G))$ dropout of $C_k(G)$, we need to delete $deg(v, C_k(G)) - k + 1$ v 's neighbors in $C_k(G)$, i.e., local perspective. However, it is undesirable to put all these neighbors of each node in \mathcal{E} into \mathcal{D} . Considering the effect of cascade in the k -core, i.e., global perspective, it may be enough to remove only a very small number of nodes from $C_k(G)$, such that the degree of each node in \mathcal{E} is less than k in $C_k(G)$. Motivated by this, for each node u in $C_k(G)$, we use $Dec(u)$ to measure the contribution of deleting u in the MDK problem. Specifically, $Dec(u)$ is the number of neighbors of the nodes in \mathcal{E} that are removed with the deletion of u , i.e.,

$$Dec(u) = \sum_{v \in \mathcal{E}} deg(v, C_k(G)) - deg(v, C_k(G \setminus \{u\}))$$

Based on the concept of $Dec(u)$, we introduce our degree decline heuristic (DDH) algorithm. The details of the algorithm is shown in Algorithm 7. In line 1, we first compute the k -core of G . Then, we iteratively select a node u^* from $C_k(G)$ with the maximum $Dec(u)$ for deletion, until all nodes in \mathcal{E} do not exist in k -core (lines 3-9). Finally, we return the set of detached nodes in line 10.

Algorithm 7: DDH Algorithm

Input : G : a graph, k : degree constraint, \mathcal{E} : conflict node set
Output : \mathcal{D} : the set of detached nodes

- 1 $G' \leftarrow C_k(G)$;
- 2 $\mathcal{D} \leftarrow \emptyset$;
- 3 **while** $\mathcal{E} \neq \emptyset$ **do**
- 4 **for each** $u \in V(G')$ **do**
- 5 compute $Dec(u)$;
- 6 $u^* \leftarrow$ the node with maximum Dec ;
- 7 $\mathcal{D} \leftarrow \mathcal{D} \cup \{u^*\}$;
- 8 $G' \leftarrow C_k(G' \setminus \{u^*\})$;
- 9 update \mathcal{E} ;
- 10 **return** \mathcal{D}

Note that, in Algorithm 7, we need to compute $\text{Dec}(u)$ for each node in candidate set. Let $\mathcal{I}(u)$ be the set of nodes that are removed from the k -core after u is deleted, i.e.,

$$\mathcal{I}(u) = V(C_k(G)) \setminus V(C_k(G \setminus u))$$

To speed up the selection of the node with the largest Dec from $V(C_k(G))$ at each iteration, we propose the following lemma to avoid the computation of Dec for some nodes.

Lemma 4 *Given two nodes u and v in G , we have $\text{Dec}(v) \leq \text{Dec}(u)$, if $v \in \mathcal{I}(u)$.*

Proof Since $v \in \mathcal{I}(u)$, it implies that $\mathcal{I}(v) \subseteq \mathcal{I}(u)$. Hence, $\text{Dec}(v) \leq \text{Dec}(u)$, and the lemma holds.

5 Experiments

In this section, comprehensive experiments are conducted over 9 datasets to evaluate the efficiency and effectiveness of proposed techniques.

5.1 Experiment setup

Algorithms In the experiments, we implement and compare the following algorithms. The first 5 algorithms is for the IAK problem.

- **Rand:** select b nodes that are not in the k -core randomly.
- **Exact-IAK:** enumerate all possible combinations with the optimal result.
- **Traditional:** obtain the node set with traditional anchored k -core model.
- **BL:** the baseline greedy algorithm, i.e., Algorithm 2.
- **KSM:** the optimized algorithm i.e., Algorithm 5.

Then, the following 3 algorithms is for the MDK problem.

- **Exact-MDK:** enumerate all possible combinations with size from 1 to $|\mathcal{E}|$ and return the optimal solution.
- **CRH:** The conflict removing heuristic algorithm, i.e., Algorithm 6.
- **DDH:** The degree decline heuristic algorithm, i.e., Algorithm 7.

Note that, for MDK problem, there is no random and traditional solutions. This is because, the results returned by random and traditional solutions may be far from satisfied due to the removal of all the nodes in conflict set \mathcal{E} .

Datasets and workloads We conduct experiments on 8 real-world networks, which are public available in Networkrepository¹ and SNAP². We also employ 1 artificial graph, which is generated by GTGraph with 500 nodes and 5000 edges. Table 2 shows the statistic details of the datasets. Note that, nodes in the original graphs have no close or conflict

¹ <http://networkrepository.com>

² <http://snap.stanford.edu>

Table 2 Statistics of datasets

Dataset	nodes	edges	d_{avg}	d_{max}
Artificial	496	3,971	16	89
Eco-mahindas	1,258	7,513	12	206
Soc-hamsterster	2,426	16,630	14	273
Email	1,005	16,064	32	345
Facebook	4,039	88,234	44	1045
Brightkite	58,228	214,078	7.4	1134
Arxiv	34,546	420,877	24.4	846
Gowalla	196,591	950,327	10	14,730
YouTube	1,134,890	2,987,624	5	28,754

properties, so we use a method similar to BFS to assign the labels to each node in the datasets. For each connected component in the original graph, we randomly select an initial node and regard it as a close, and each neighbor of the initial node has a 50% probability of being close, a 20% probability of being conflict, and a 30% probability of being unchanged (i.e., no label). For an unchanged node, each of its neighbor is set as unchanged. Then, we repeat the process for the encountered nodes in BFS. In our experiment, both k and b vary from 5 to 25. For each setting, we run the algorithm 20 times and take the average performance as the final result. All the programs are implemented in standard C++. All the experiments are performed on a PC with an Intel i5-9600KF 3.7GHz CPU and 64 GB main memory.

5.2 Effectiveness evaluation for IAK problem

To evaluate the effectiveness of algorithms for IAK problem, we report the ratio of inclined score of KSM and Exact-IAK by anchoring b nodes. This is because KSM only enhances the efficiency compared to BL. Then, we compare the results obtained by the traditional anchored k -core and our inclined anchored k -core problems. In addition, case studies are also conducted.

5.2.1 Compared with Exact-IAK

Table 3 reports the inclined score ratio of KSM and Exact-IAK by anchoring b nodes, where $k = 10$ and b varies from 1 to 3. We set the inclined score ratio of Exact-IAK as 100% and that of KSM is $\frac{\text{score returned by KSM}}{\text{score returned by Exact-IAK}} \times 100\%$. Note that, due to the high compu-

Table 3 Inclined score ratio of Exact-IAK and KSM

Ratio	$b = 1$		$b = 2$		$b = 3$	
	Exact-IAK	KSM	Exact-IAK	KSM	Exact-IAK	KSM
Artificial	100%	100%	100%	100%	100%	100%
Eco-mahindas	100%	100%	100%	100%	100%	100%
Soc-hamsterster	100%	100%	100%	100%	100%	97%

Table 4 Response time of Exact-IAK and KSM

Running Time	$b = 1$		$b = 2$		$b = 3$	
	Exact-IAK	KSM	Exact-IAK	KSM	Exact-IAK	KSM
Artificial	0.0005	0.00005	0.1131	0.0001	18.44	0.0002
Eco-mahindas	0.0097	0.0006	5.8537	0.0011	2510	0.0016
Soc-hamsterster	0.0435	0.0001	50.913	0.0003	42985.6	0.0005

tational cost of Exact-IAK, we only test Exact-IAK on small datasets with small b values. As observed, KSM only slightly drops in one setting. In addition, Table 4 reports the running time of Exact-IAK and KSM. We can see KSM achieves significant speedup compared with Exact-IAK, which further verifies the effectiveness of the greedy framework.

5.2.2 Compared with rand and traditional

Compared with Rand In Figure 4(a), we report the inclined score compared with Rand on 6 larger datasets, i.e., Email, Facebook, Brightkite, Arxiv, Gowalla, YouTube with $b = 10$ and $k = 10$. The datasets are ordered by their network sizes, i.e., the number of edges. Note that, the larger the inclined score is, the more effective the algorithm will be. As we can see, KSM outperforms Rand by a big margin. This is because, there is usually few followers for anchored nodes that are selected in random.

Compared with Traditional We report the results by compared the inclined model with the traditional model. The corresponding inclined scores are shown in Figure 4(b). Traditional does not consider the properties of different nodes, and only focuses on enlarging the total k -core. As observed, Traditional may lead to a very small inclined score, even a negative score, such as Youtube. Thus, it is necessary to develop algorithms to handle the inclined case.

5.2.3 Case study

We conduct a case study on Facebook dataset with $k = 20$ and $b = 1$. Figure 5(a) and (b) are the results obtained by KSM and Traditional, respectively, where blue/red nodes denotes the close/conflict nodes, and white nodes are the ones without labels. As shown in Figure 5(a), the best anchored node is the one with id 2300. It has 5 followers and all of them are close nodes, i.e., the inclined score is 5. As shown in Figure 5(b),

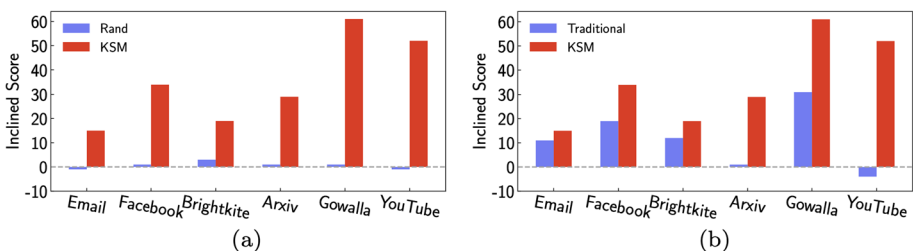


Figure 4 Effectiveness evaluation compared with Rand and Traditional for IAK problem

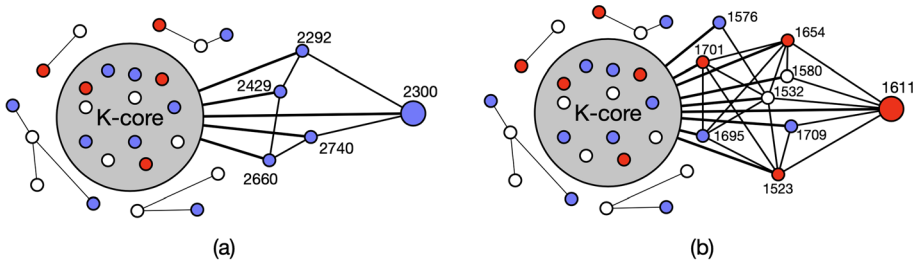


Figure 5 Case study for IAK problem

the returned anchor is with id 1611. It has 9 followers, where there are 3 close and 4 conflict nodes, i.e., the inclined score is -1. Compared with the anchored node 2300, though the node 1611 has more followers, it has smaller inclined score.

5.3 Effectiveness evaluation for MDK problem

To evaluate the effectiveness of the proposed two heuristic algorithms for MDK problem, we compare them with Exact-MDK on a small dataset. Then, we conduct experiments on the large datasets to further evaluate the performance of the two strategies.

5.3.1 Compare heuristic solutions with Exact-MDK

Due to the prohibitive time cost for Exact-MDK algorithm, in this experiment, we select a small dataset, i.e., Eco-mahindas, to evaluate the performance of the heuristic solutions and Exact-MDK by varying k and $|\mathcal{E}|$. The results are shown in Figure 6. The number on the bar is the corresponding response time. Since both heuristics take less than 10^{-3} seconds to complete, we only mark the time cost for Exact-MDK in the figure. As we can see, the two heuristics perform as well as the Exact-MDK in most settings. Moreover, the heuristic algorithms are much faster than Exact-MDK, which verifies the advantage of proposed heuristic strategies.

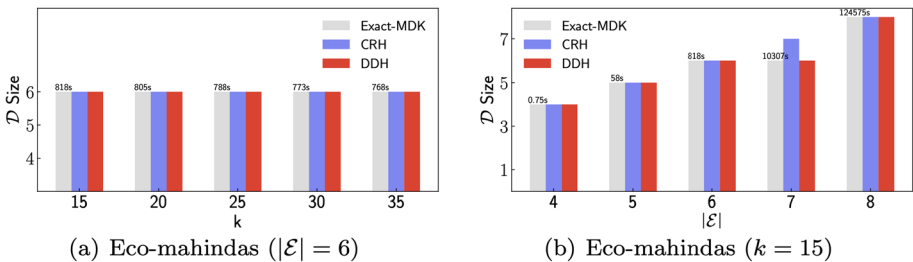


Figure 6 Effectiveness evaluation compared with Exact-MDK

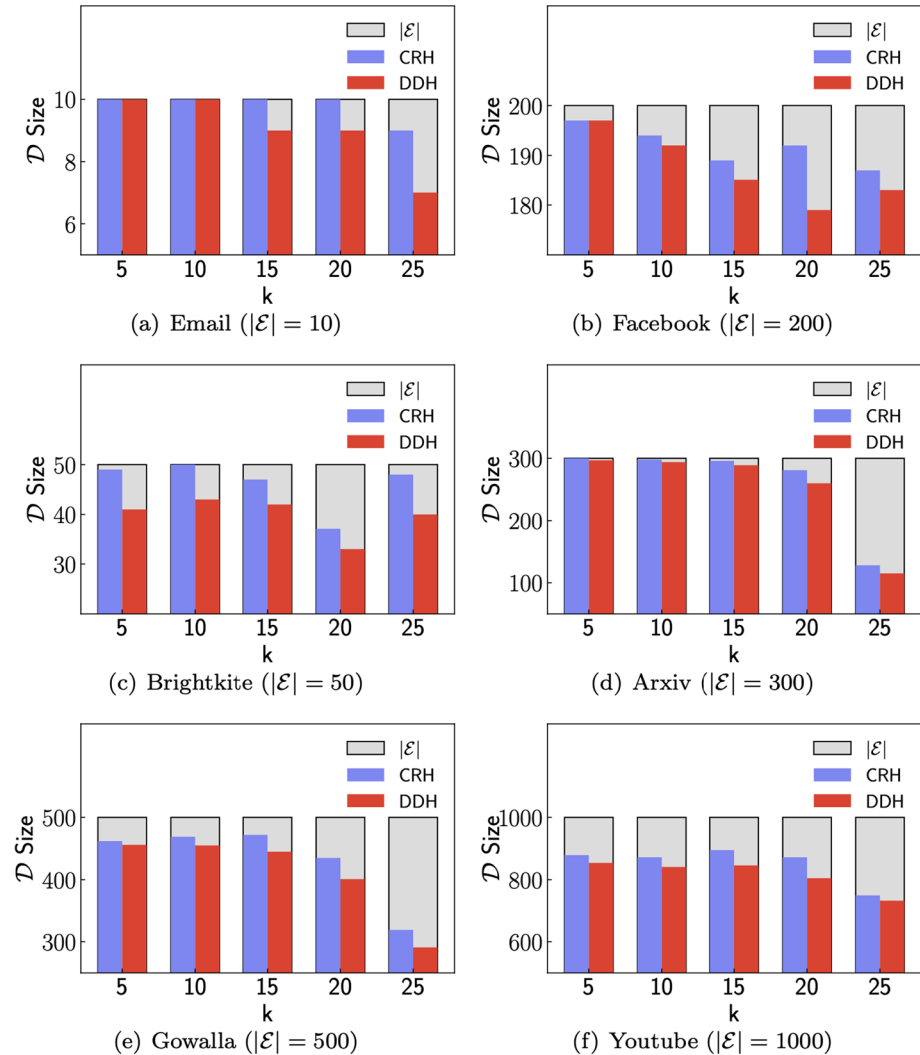


Figure 7 Compare the size of detached node set by varying k

5.3.2 Compare detached node set size

In this experiment, we report the size of detached node set, i.e., $|\mathcal{D}|$, retrieved by CRH and DDH in Figures 7 and 8 by varying k and the number of conflict nodes on the largest 6 datasets, respectively. Recall that, $\mathcal{D} = \mathcal{E}$ is an answer of MDK problem. Therefore, we use $|\mathcal{E}|$ as a benchmark to measure the effectiveness of CRH and DDH, i.e., the gray bar in Figures 7 and 8. As we have seen, in most cases, both CRH and DDH are able to obtain smaller \mathcal{D} . DDH performs better than CRH, because CRH is able to select nodes from global perspective, i.e., leveraging the cascade phenomenon of k -core. While, DDH is based on local observation, i.e., only replace part of nodes in $\mathcal{D} = \mathcal{E}$. For example, in

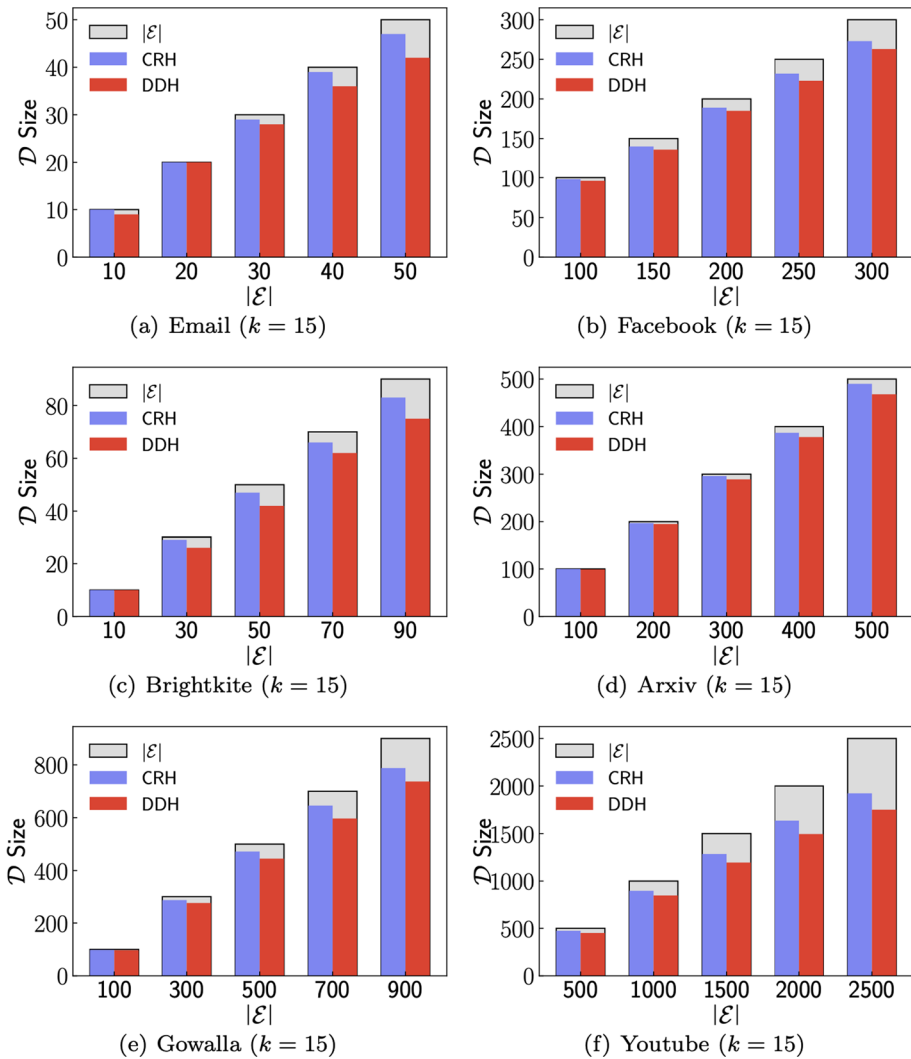


Figure 8 Compare the size of detached node set by varying $|\mathcal{E}|$

Figure 7(d) with $k = 25$, $|\mathcal{E}|$ is 300, while the size of \mathcal{D} retrieved by CRH and DDH are 128 and 115, respectively. The results of CRH and DDH are similar only when k or $|\mathcal{E}|$ is small. This is because, the cascading phenomenon becomes more prominent for larger k and $|\mathcal{E}|$.

5.3.3 Compare size of remaining k -core

In this experiment, we report the size of remaining k -core, i.e., $|C_k(G \setminus \mathcal{D})|$, after the deletion of detached node set from graph G for CRH and DDH. The result are shown in Figures 9 and 10 by varying k and the number of conflict nodes on the largest 6 datasets, respectively. We use the size of original k -core as a benchmark, i.e., $|C_k(G)|$. It can be seen from the results

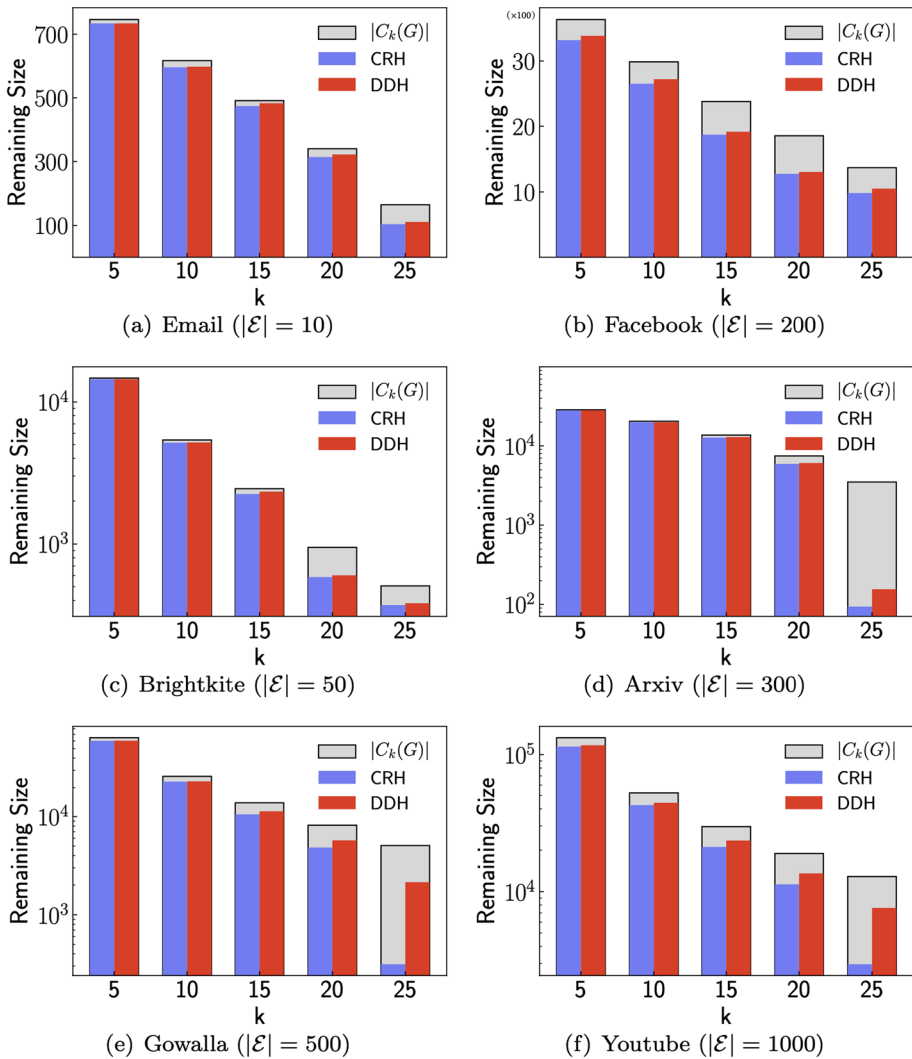


Figure 9 Compare the size of remaining k -core by varying k

that the remaining k -core becomes smaller due to the removal of \mathcal{D} , and the variation increases gradually with the increase of k and $|\mathcal{E}|$. Although DDH takes into account the cascade effect when selecting nodes, the final remaining k -core is usually larger than that of CRH. For example, in Figure 9(e) with $k = 25$, the size of remaining k -core of DDH is 2138, while CRH is only 313. From this perspective, it also reflects that DDH is more effective than CRH.

5.4 Efficiency evaluation for IAk problem

To evaluate the efficiency of our algorithms for IAk problem, we report the response time of BL and KSM by varying b and k on the largest 6 datasets. The results are shown in Figures 11 and 12, respectively. It is clear that KSM constantly outperforms BL on all datasets, and can

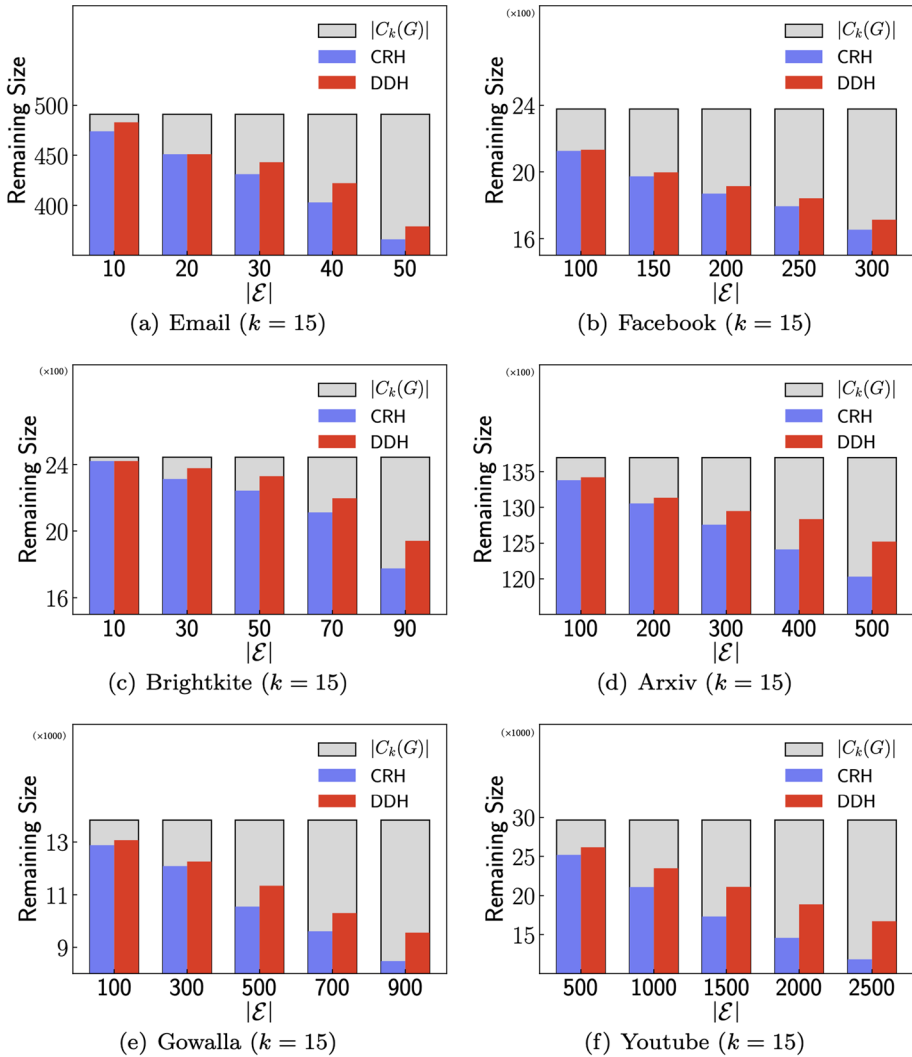


Figure 10 Compare the size of remaining k -core by varying $|\mathcal{E}|$

achieve up to 2 orders of magnitude speedup. As observed, the response time of both methods increases when b increases. This is because more iterations need to be conducted. When k increases, the response time also grows. This is because when k is larger, the nodes in the layer structure have a larger degree, which leads to the need to explore more neighbors and greater time consumption.

5.5 Efficiency evaluation for MDK problem

In this experiment, we report the response time of CRH and DDH algorithms for MDK problem by varying k and the size of conflict node set on the largest 6 datasets. The results are

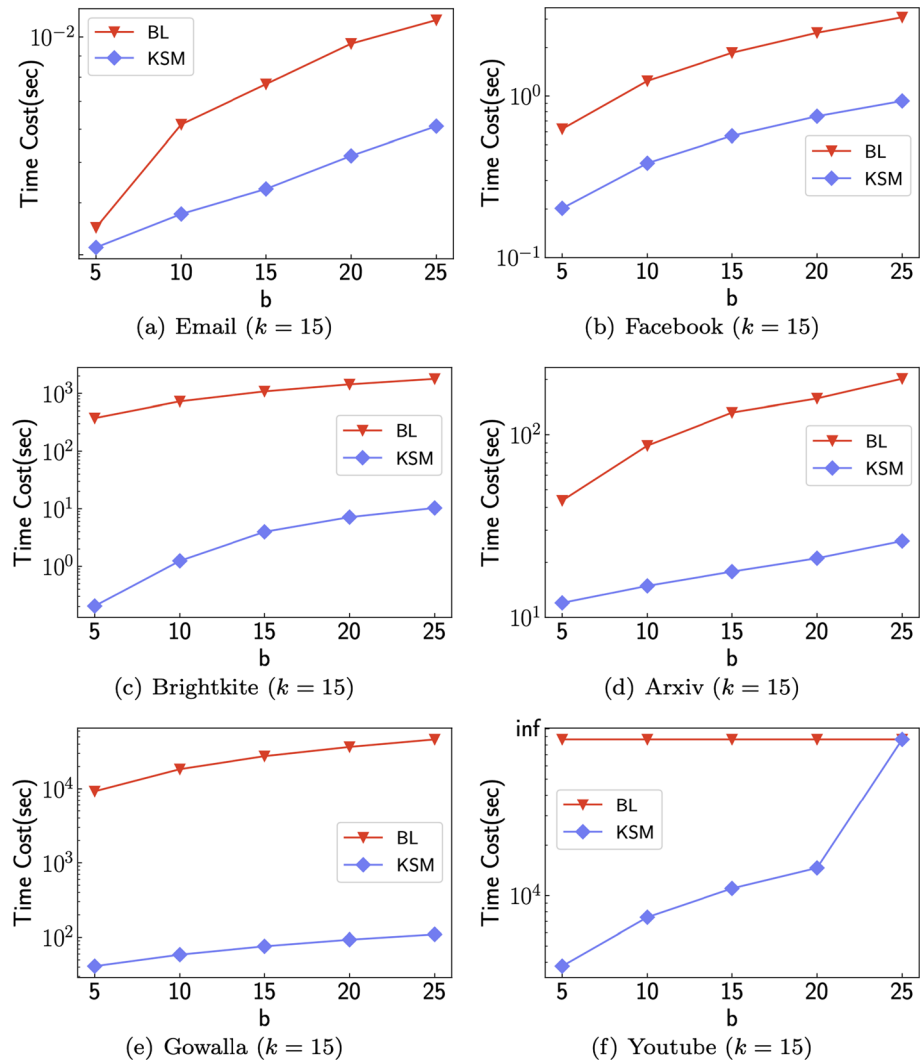


Figure 11 Efficiency evaluation for IAK problem by varying b

shown in Figures 13 and 14. As can be seen from the results, CRH is basically faster than DDH by at least one order of magnitude. This is because CRH only needs to visit the nodes in \mathcal{E} and their neighbors, i.e., local observation, while DDH needs to traverse all the nodes in $C_k(G)$. However, DDH is also able to finish in a competitive time, and takes only about 100 seconds for the largest graph. Since CRH only needs to traverse \mathcal{E} , the time required for CRH is essentially constant when k varies. Moreover, the response time CRH slightly increases when the size of \mathcal{E} becomes larger. The response time of DDH, on the other hand, is sensitive to the change of k and $|\mathcal{E}|$. With the increase of k , DDH runs faster, because there are fewer nodes in k -core. However, as $|\mathcal{E}|$ increases, the response time of DDH become larger. This is because it takes more iterations to exclude the nodes in \mathcal{E} from k -core.

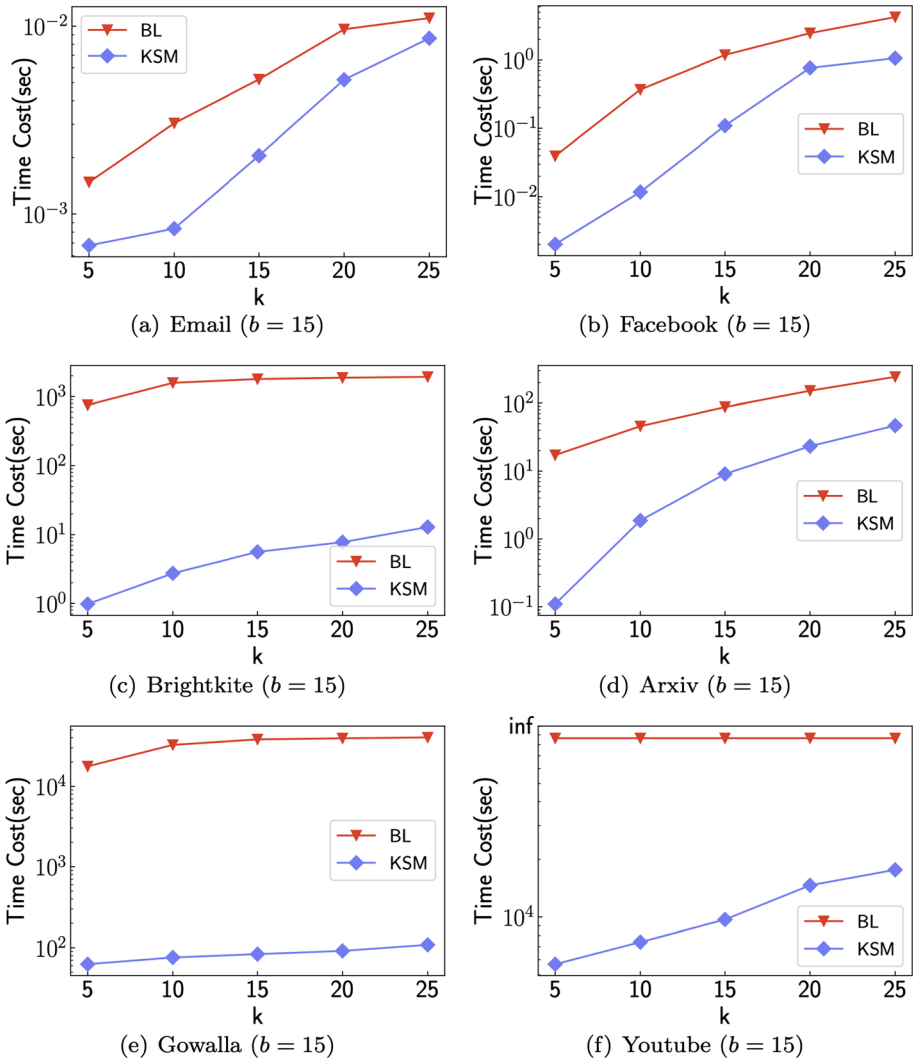


Figure 12 Efficiency evaluation for IAK problem by varying k

6 Related work

As a common data structure, graphs are widely used to model the relationships among different entities, such as IoT [14], social network [27] and brain network [24]. In social network analysis, different cohesive subgraph models have been proposed to accommodate different scenarios, such as k -core [20, 21], k -truss [13, 30] and clique [5, 23]. The k -core model is firstly proposed by Seidman [20], which has been widely adopted for social network analysis with numerous applications, such as protein function prediction [26], social contagion [25] and influence study [16].

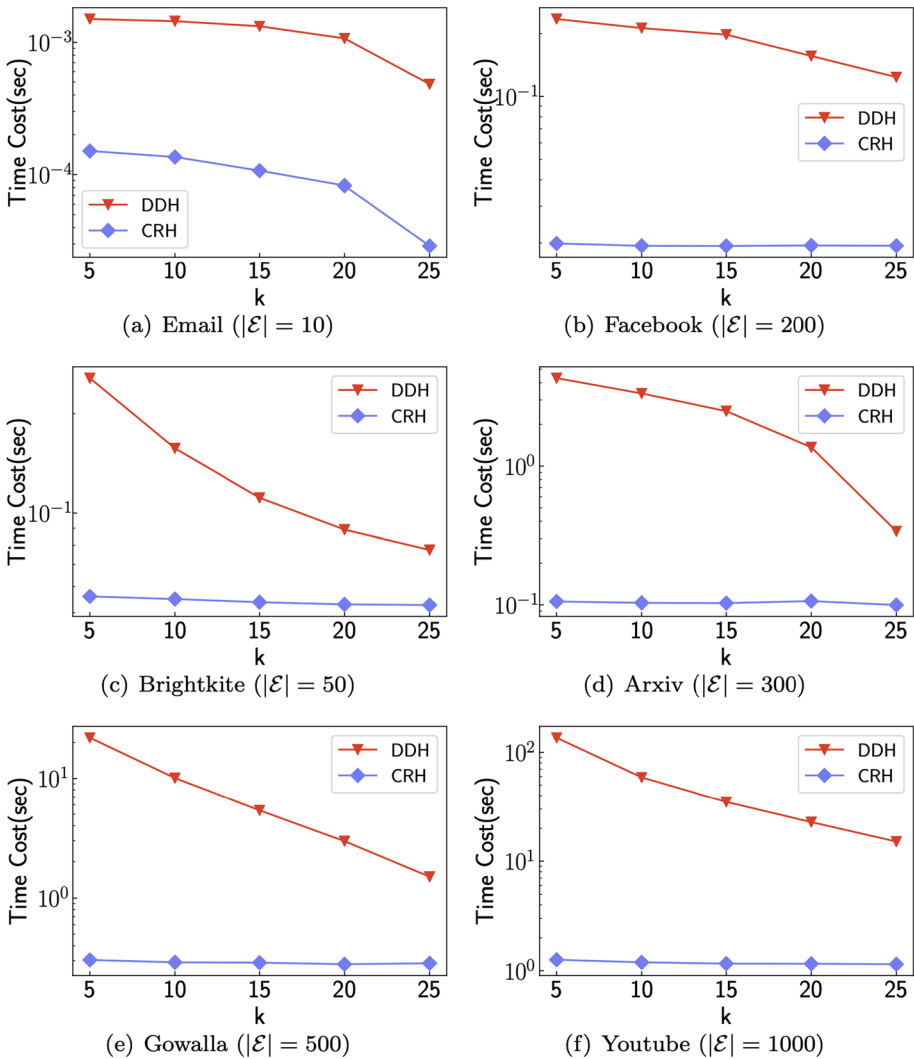


Figure 13 Efficiency evaluation for MDK problem by varying k

In social networks, the engagement and breakdown of certain nodes/edges can greatly influence the corresponding community. Due to the unique applications, the problems of finding critical nodes or edges have attracted significant attention in the recent, and different cohesive subgraph models and settings are studied, e.g., [2, 8, 28, 29]. These problems are usually NP-hard. Therefore, heuristic strategies are widely adopted in the previous studies. Bhawalkar et al. [2] propose the anchored k -core problem, which aims to maximize the size of k -core by anchoring b nodes. The authors also prove the problem is NP-hard. To accelerate the computation, [28] develops an efficient algorithm for the anchored k -core problem on large-scale graphs. [11] proposes the directed anchored k -core problem by considering the case in directed graphs. In [17], authors further investigate the anchored k -core problem by minimizing the total budget. Different from the anchored k -core problem, the collapsed k -core problem aims to minimize the size of k -core community by deleting certain

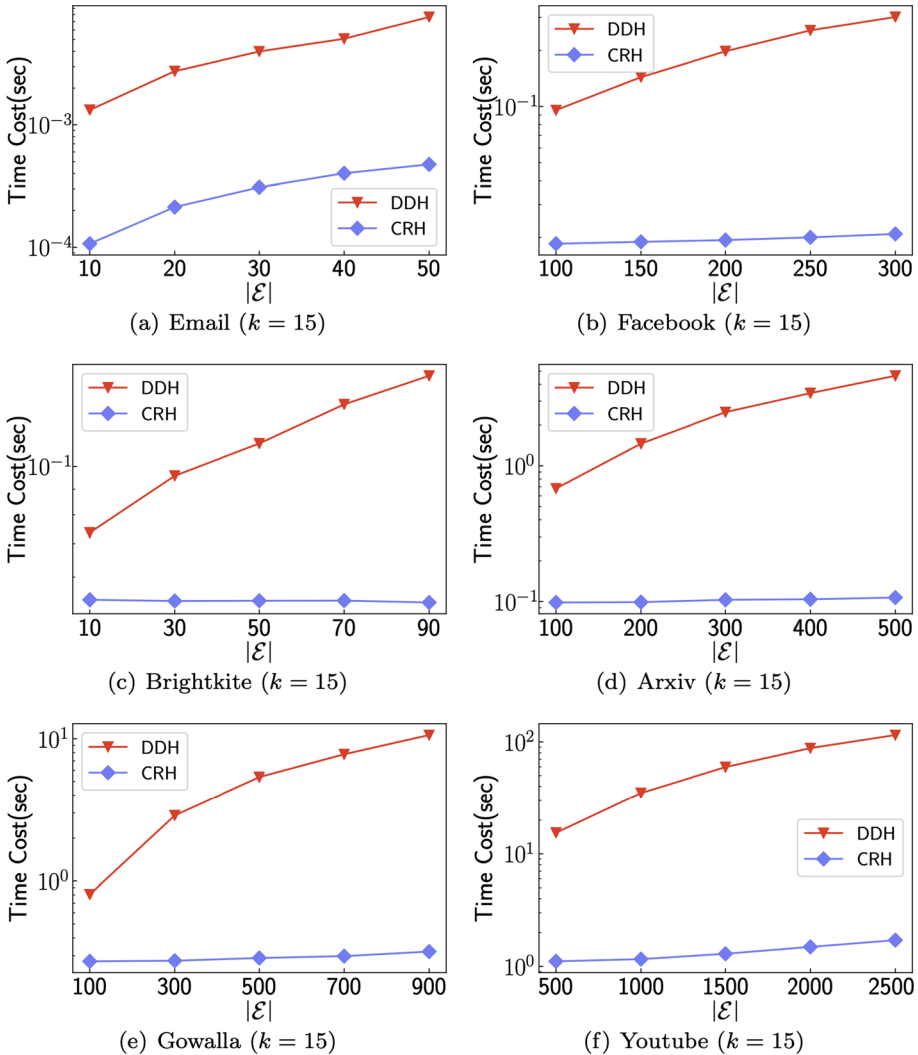


Figure 14 Efficiency evaluation for MDK problem by varying $|\mathcal{E}|$

nodes [29]. In [7, 32], authors attempt to minimize the corresponding k -core by removing the critical edges from the graph. [18, 19] consider the k -core minimization problem based on the game theory model. Besides k -core model, the collapsed problems are also investigated under different models, such as k -truss model [9, 33] and bipartite graph settings [31]. However, none of them considers the different property of nodes, i.e., close or conflict.

7 Conclusion

Anchored/collapsed k -core problem has attracted great attentions in the recent. However, the previous studies usually treat the nodes equally, which may fail to better characterize

the real scenarios. Motivated by this, in this paper, we propose and study the inclined anchored k -core and minimum detached k -core problems by considering the node properties, i.e., close or conflict. We formally define the problem and prove their hardness. To facilitate the computations, different search methods and heuristic strategies are developed. Finally, comprehensive experiments on real-life networks are conducted to verify the effectiveness and efficiency of the developed techniques.

Acknowledgements The paper is a journal extension of our WISE 2021 conference paper [4]. This work was supported by NSFC 61802345, ZJNSF LQ20F020007, ZJNSF LY21F020012 and Y202045024.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.


References

1. Batagelj, V., Zaversnik, M.: An $o(m)$ algorithm for cores decomposition of networks. *CoRR* (2003)
2. Bhawalkar, K., Kleinberg, J., Lewi, K., Roughgarden, T., Sharma, A.: Preventing unraveling in social networks: the anchored k -core problem. *SIAM Journal on Discrete Mathematics* **29**(3), 1452–1475 (2015)
3. Burke, M., Marlow, C., Lento, T.: Feed me: motivating newcomer contribution in social network sites. In: *SIGCHI*, pp. 945–954 (2009)
4. Chen, C., Liu, X., Xu, S., Zhang, M., Wang, X., Lin, X.: Critical nodes identification in large networks: An inclination-based model. In: *WISE*, pp. 453–468 (2021)
5. Chen, C., Wu, Y., Sun, R., Wang, X.: Maximum signed θ -clique identification in large signed graphs. *TKDE* (2021)
6. Chen, C., Zhang, M., Sun, R., Wang, X., Zhu, W., Wang, X.: Locating pivotal connections: The k -truss minimization and maximization problems. *WWW Journal* pp. 1–28 (2021)
7. Chen, C., Zhu, Q., Sun, R., Wang, X., Wu, Y.: Edge manipulation approaches for k -core minimization: Metrics and analytics. *TKDE* (2021)
8. Chen, C., Zhu, Q., Wu, Y., Sun, R., Wang, X., Liu, X.: Efficient critical relationships identification in bipartite networks. *WWW Journal* (2021)
9. Chen, H., Conte, A., Grossi, R., Loukides, G., Pissis, S.P., Sweering, M.: On breaking truss-based communities. In: *KDD*, pp. 117–126 (2021)
10. Cheng, D., Chen, C., Wang, X., Xiang, S.: Efficient top- k vulnerable nodes detection in uncertain graphs. *TKDE* (2021)
11. Chitnis, R., Fomin, F.V., Golovach, P.A.: Parameterized complexity of the anchored k -core problem for directed graphs. *Information and Computation* **247**, 11–22 (2016)
12. Chitnis, R.H., Fomin, F.V., Golovach, P.A.: Preventing unraveling in social networks gets harder. In: *AAAI* (2013)
13. Cohen, J.: Trusses: Cohesive subgraphs for social network analysis. National security agency technical report (2008)
14. He, J., Rong, J., Sun, L., Wang, H., Zhang, Y., Ma, J.: A framework for cardiac arrhythmia detection from IoT-based ECGs. *World Wide Web* **23**(5), 2835–2850 (2020)
15. Karp, R.M.: Reducibility among combinatorial problems. In: *Complexity of computer computations*, pp. 85–103. Springer (1972)
16. Kitsak, M., Gallos, L.K., Havlin, S., Liljeros, F., Muchnik, L., Stanley, H.E., Makse, H.A.: Identification of influential spreaders in complex networks. *Nature physics* **6**(11), 888–893 (2010)
17. Liu, K., Wang, S., Zhang, Y., Xing, C.: An efficient algorithm for the anchored k -core budget minimization problem. In: *ICDE*, pp. 1356–1367 (2021)
18. Medya, S., Ma, T., Silva, A., Singh, A.: A game theoretic approach for core resilience. In: *IJCAI* (2020)
19. Medya, S., Ma, T., Silva, A., Singh, A.: A game theoretic approach for k -core minimization. In: *International Conference on Autonomous Agents and MultiAgent Systems* (2020)
20. Seidman, S.B.: Network structure and minimum degree. *Social networks* **5**(3), 269–287 (1983)

21. Sun, R., Chen, C., Wang, X., Wu, Y., Zhang, M., Liu, X.: The art of characterization in large networks: Finding the critical attributes. *WWW Journal* (2021)
22. Sun, R., Chen, C., Wang, X., Zhang, Y., Wang, X.: Stable community detection in signed social networks. *TKDE* (2020)
23. Sun, R., Zhu, Q., Chen, C., Wang, X., Zhang, Y., Wang, X.: Discovering cliques in signed networks based on balance theory. In: *DASFAA* (2020)
24. Tawhid, Md. N.A., Siuly S., Wang, K., Wang, H.: Data mining based artificial intelligent technique for identifying abnormalities from brain signal data: *WISE*, 198–206 (2021)
25. Ugander, J., Backstrom, L., Marlow, C., Kleinberg, J.: Structural diversity in social contagion. *Proceedings of the National Academy of Sciences* **109**(16), 5962–5966 (2012)
26. Vazquez, A., Flammini, A., Maritan, A., Vespignani, A.: Global protein function prediction from protein-protein interaction networks. *Nature biotechnology* **21**(6), 697–700 (2003)
27. Wang, X., Zhang, Y., Zhang, W., Lin, X., Chen, C.: Bring order into the samples: A novel scalable method for influence maximization. *IEEE Transactions on Knowledge and Data Engineering* **29**(2), 243–256 (2016)
28. Zhang, F., Zhang, W., Zhang, Y., Qin, L., Lin, X.: Olak: an efficient algorithm to prevent unraveling in social networks. *PVLDB* **10**(6), 649–660 (2017)
29. Zhang, F., Zhang, Y., Qin, L., Zhang, W., Lin, X.: Finding critical users for social network engagement: The collapsed k-core problem. In: *AAAI* (2017)
30. Zhao, J., Sun, R., Zhu, Q., Wang, X., Chen, C.: Community identification in signed networks: a k-truss based model. In: *CIKM* (2020)
31. Zhu, Q., Zheng, J., Yang, H., Chen, C., Wang, X., Zhang, Y.: Hurricane in bipartite graphs: The lethal nodes of butterflies. In: *SSDBM* (2020)
32. Zhu, W., Chen, C., Wang, X., Lin, X.: K-core minimization: An edge manipulation approach. In: *CIKM*, pp. 1667–1670 (2018)
33. Zhu, W., Zhang, M., Chen, C., Wang, X., Zhang, F., Lin, X.: Pivotal relationship identification: The k-truss minimization problem. In: *IJCAI* (2019)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Renjie Sun¹ · Chen Chen²  · Xijuan Liu² · Shuangyan Xu² · Xiaoyang Wang² · Xuemin Lin³

Renjie Sun
renjie.sun@stu.ecnu.edu.cn

Xijuan Liu
liuxijuan@zjgsu.edu.cn

Shuangyan Xu
xshy@zjgsu.edu.cn

Xiaoyang Wang
xiaoyangw@zjgsu.edu.cn

Xuemin Lin
lxue@cse.unsw.edu.au

¹ East China Normal University, Shanghai, China

² Zhejiang Gongshang University, Hangzhou, China

³ University of New South Wales, Sydney, Australia