K-Core Maximization: An Edge Addition Approach

Zhongxin Zhou[1,2,4], **Fan Zhang**[1], **Xuemin Lin**[2,3,4], **Wenjie Zhang**[3] and **Chen Chen**[2]
[1]Guangzhou University, Guangzhou, China
[2]East China Normal University, Shanghai, China
[3]University of New South Wales, Sydney, Australia
[4]Zhejiang Lab, Hangzhou, China
{zzxecnu, fanzhang.cs}@gmail.com, {lxue, zhangw}@cse.unsw.edu.au, chenc@zjgsu.edu.cn

Abstract

A popular model to measure the stability of a network is k-core - the maximal induced subgraph in which every vertex has at least k neighbors. Many studies maximize the number of vertices in k-core to improve the stability of a network. In this paper, we study the edge k-core problem: Given a graph G, an integer k and a budget b, add b edges to non-adjacent vertex pairs in G such that the k-core is maximized. We prove the problem is NP-hard and APX-hard. A heuristic algorithm is proposed on general graphs with effective optimization techniques. Comprehensive experiments on 9 real-life datasets demonstrate the effectiveness and the efficiency of our proposed methods.

1 Introduction

Graphs are widely used to model networks, where each vertex represents a user and each edge represents a connection between two users. The cohesive subgraph model of k-core, introduced by [Seidman, 1983], is defined as the maximal induced subgraph in which every vertex has at least k neighbors (adjacent vertices) in the subgraph. The k-core of a network corresponds to the natural equilibrium of a user engagement model: each user incurs a cost (e.g., k) to remain engaged but receives a benefit proportional to (e.g., equal to) the number of engaged neighbors. Since the number of vertices (the size) of k-core reflects the stability of a network, it is widely adopted in the study of network engagement (stability), e.g., [Bhawalkar *et al.*, 2015; Malliaros and Vazirgiannis, 2013; Wu *et al.*, 2013].

To prevent network unraveling, Bhawalkar and Kleinberg et al. propose the anchored k-core problem which maximizes the k-core by anchoring b vertices [Bhawalkar *et al.*, 2015], where the degree of an anchor is considered as infinitely large. There are a series of following work to maximize the k-core, e.g., [Zhang *et al.*, 2018b; Zhang *et al.*, 2017a; Chitnis *et al.*, 2013]. In order to improve network stability, another basic graph operation is edge addition, which can also be applied to k-core maximization. Thus, the edge k-core problem is proposed [Chitnis and Talmon, 2018]: Given a graph G, an integer k and a budget b, add b edges to non-adjacent vertex pairs in G such that the k-core is the largest.

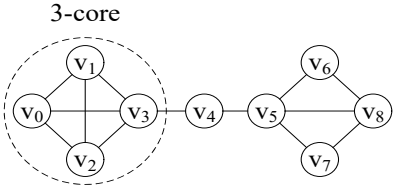


Figure 1: An Example of k-Core and Anchoring, $k = 3$

Example 1. *Figure 1 depicts a social group G with 9 users and their connections. The willingness of a user to keep engaged is influenced by the number of her friends (neighbors) in this group. According to the k-core model, suppose $k = 3$, v_4, v_6, and v_7 firstly drop out. Their departure leads to the leave of v_5 and v_8, as their degrees decrease to 1 which is less than k. To improve network stability, we can anchor v_6 and v_7 based on anchored k-core model, or add an edge between v_6 and v_7 based on edge k-core model. Both solutions lead to a larger k-core induced by the vertices in G except v_4.*

The edge k-core problem can find many applications on real-life networks: friend recommendation in social networks, connection construction in telecom networks, etc. For instance, in a P2P network, any user benefiting from the network should be connected to at least k other users, to exchange resources. The holder of a P2P network can use the edge k-core model to find which connections should be added between users so that a large number of users can successfully use the P2P network [Chitnis and Talmon, 2018].

Challenges and Contributions. In this paper, we propose a concise reduction to prove that the problem is NP-hard and APX-hard. The only existing solution is proposed for graphs with bounded tree-width [Chitnis and Talmon, 2018]. However, this assumption usually does not hold in real-life graphs, and their techniques cannot be extended to handle general graphs. Due to the hardness of the problem, we propose a heuristic algorithm with effective pruning techniques. The experiments are conducted on 9 real networks to demonstrate the effectiveness and the efficiency of the proposed methods.

Zhongxin Zhou and Fan Zhang are the joint first authors. Fan Zhang is the corresponding author.

2 Related Work

Graph processing on large data may require higher computation efficiency than traditional queries [Luo *et al.*, 2008; Cheema *et al.*, 2010; Luo *et al.*, 2011]. Cohesive subgraph mining is a fundamental graph problem, with various models such as clique [Luce and Perry, 1949], k-core [Seidman, 1983], k-fami [Zhang *et al.*, 2018a], etc. Among the models, k-core is the only one known to have a linear time algorithm [Batagelj and Zaversnik, 2003]. The k-core has a wide range of applications such as social contagion [Ugander *et al.*, 2012], influential spreader identification [Kitsak *et al.*, 2010], collapse prediction [Morone *et al.*, 2019], user engagement study [Malliaros and Vazirgiannis, 2013], etc.

There is an efficient heuristic algorithm for the anchored k-core problem [Zhang *et al.*, 2017a], while it cannot be simply applied to solve the edge k-core problem. One major reason is that the anchored k-core model does not change the topology of the graph while the edge k-core model needs to add new edges. Besides the k-core maximization work introduced in Section 1, there are some studies on k-core minimization under the view of against attack [Zhang *et al.*, 2017b; Zhu *et al.*, 2018; Medya *et al.*, 2019].

The edge addition has been studied in different topics. [Natanzon *et al.*, 2001] studies the hardness of edge modification problems on some classes of graphs. [Suady and Najim, 2014] aims to reduce the diameter of a graph by adding edges. [Lai *et al.*, 2005] aims to add a small number of edges in a graph to enlarge the bandwidth. [Kapron *et al.*, 2011] aims to anonymize a given vertex set by adding fewest edges.

3 Preliminaries

We consider a simple, undirected and unweighted graph $G = (V, E)$, where V is a set of vertices and and E is a set of edges. We denote $n = |V|$, $m = |E|$ and assume $m > n$. Let $S = (V', E')$ be an induced subgraph of G, where $V' \subseteq V$ and $E' \subseteq E$. The notations are summarized in Table 1.

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

The k-core of a graph G can be obtained by recursively removing every vertex u and its incident edges in G if $deg(u, G) < k$, with a time complexity of $O(m)$. The k-cores of G with different inputs of k constitute a hierarchical structure of G, i.e., $C_{k+1}(G) \subseteq C_k(G)$ for every value of k. The definition of k-shell is then derived.

Definition 2. ***k*-shell**. *Given a graph G, the k-shell of G, denoted by $H_k(G)$, is the set of vertices in k-core but not in (k+1)-core, i.e., $H_k(G) = V(C_k(G) - C_{k+1}(G))$.*

If we add some new edges among the vertices which are not adjacent, the k-core of the graph may contain more vertices, which is named the edge k-core. The added new edges are called anchors or anchor edges. In this paper, we say anchor, add, or insert an edge interchangeably, e.g., an inserted edge is also called an anchored edge. The edges, which may be inserted to the graph, are called candidate anchors or candidate edges.

Notation	Definition
G	an unweighted and undirected graph
u, v; e, (u, v)	a vertex in G; an edge in G
m; n	the number of edges in G; the number of vertices in G
$N(u, G)$	the set of adjacent vertices (neighbors) of u in G
$deg(u, G)$	the number of adjacent vertices of u in G
S	a subgraph of G
$V(S)$; $E(S)$	the vertex set of S; the edge set of S
$G[X]$	induced subgraph of the vertex set X in G
$C_k(G)$; $H_k(G)$	the k-core of G; the k-shell of G
k; b	the degree constraint; the anchor budget
A	a set of anchor edges
G_A; G_e	the graph $G + A$; the graph $G + \{e\}$
$\mathcal{F}(A, G)$	the followers of the anchor set A in G
$\mathcal{L}$; L_i	the onion layers of G; the i-th layer of $\mathcal{L}$
$l(u)$	layer index of u in $\mathcal{L}$
$d^*(u)$	number of neighbors of u in its higher layers

Table 1: Summary of Notations

Definition 3. ***edge* *k*-core**. *Given a graph G and a set of anchor edges $A \subseteq (\binom{V}{2} \setminus E)$, the edge k-core, denoted by $C_k(G + A)$, is the k-core of the graph $G' = (V, E \cup A)$.*

Due to the addition of anchor edges (A), more vertices might be retained in $C_k(G + A)$, in addition to the vertices in $C_k(G)$. Note that the vertices not incident to the anchor edges may also be retained, according to the contagious nature of k-core computation. The vertices following the anchor edges A to engage in k-core are named the followers of A, denoted by $\mathcal{F}(A, G)$. Formally, $\mathcal{F}(A, G)$ is the set of vertices in $C_k(G + A) \setminus C_k(G)$. The number of the followers reflects the importance of the corresponding anchor edges.

Problem Statement. Given a graph G, a degree constraint k and a budget b, the edge k-core problem aims to find a set A of b edges in $\binom{V}{2} \setminus E$ such that the number of followers of A is maximized, i.e., $\mathcal{F}(A, G)$ is maximized.

4 Complexity

Theorem 1. *Edge k-core problem is NP-hard when $k \geq 3$.*

Proof. We reduce the edge k-core problem from the maximum coverage (MC) problem [Karp, 1972] which is NP-hard. The MC problem is to find at most b sets to cover the largest number of elements, where b is a given budget. We consider an arbitrary instance of MC with c sets $T_1, .., T_c$ and d elements $\{e_1, .., e_d\} = \cup_{1 \leq i \leq c} T_i$. We suppose $c > k$, $c > b$, and each of the resulting b sets contains at least 2 elements, without loss of generality. Then we construct a corresponding instance of the edge k-core problem on a graph G. Figure 2 shows a construction example from 3 sets and 4 elements when $k = 3$.

The set of vertices in G consists of three parts: M, N and Q. The part M contains c set of vertices where each set has $d + 3$ vertices, i.e., $M = \cup_{1 \leq i \leq c} M_i$ where $M_i = \cup_{1 \leq j \leq d+3} u_j^i$. The part N contains d vertices. The part Q is a (k+1)-clique where every two vertices of the $k + 1$ vertices

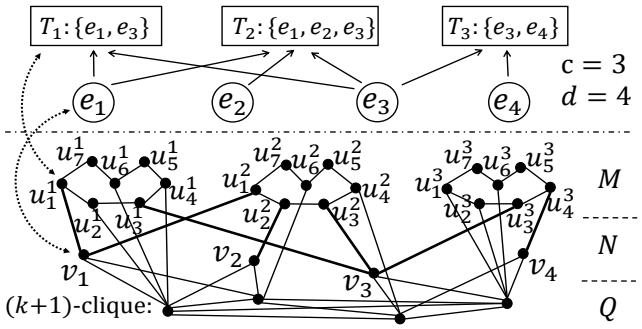


Figure 2: Complexity Reduction, $k = 3$

are adjacent. For every i and j, if $e_i \in T_j$ in the MC instance, we add an edge between v_i and u_i^j. In Figure 2, these edges are marked in bold. For every M_i in M, we connect u_j^i and u_{j+1}^i by an edge for every $j \in [1, d+2]$, and we also connect u_1^i and u_{d+3}^i. For every M_i, we add edges between every vertex in M_i and the vertices in Q so that every vertex in $M_i \setminus \{u_{d+1}^i, u_{d+3}^i\}$ has a degree of k and every vertex in $\{u_{d+1}^i, u_{d+3}^i\}$ has a degree of $k-1$. We add $k-1$ edges between v_i and the vertices in Q for every $i \in [1, d]$. The construction of G is completed.

The k-core computation will delete all the vertices in M and N. Thus, the k-core of G is Q. To enlarge the k-core, the solution (A) is to add edges between u_{d+1}^i and u_{d+3}^i, which is most cost-effective. Another solution is to add edges between M_i and M_j where $i \neq j$ may also enlarge the k-core, while this solution can always be replaced by solution (A) with at most the same number of followers. Adding an edge to a vertex in N is not worthwhile, since we suppose each of the resulting b sets contains at least 2 elements. Thus, the edge k-core problem always chooses b of the M_i in G which corresponds to b sets in the MC problem. If there is a polynomial time solution for the edge k-core problem, the MC problem will be solved in polynomial time. □

Theorem 2. *For $k \geq 3$ and any $\epsilon > 0$, the edge k-core problem cannot be approximated in polynomial time within a ratio of $(1 - 1/e + \epsilon)$, unless $P = NP$.*

Proof. We reduce from the MC problem using a reduction similar to that in the proof of Theorem 1. For any $\epsilon > 0$, the MC problem cannot be approximated in polynomial time within a ratio of $(1 - 1/e + \epsilon)$, unless $P = NP$ [Feige, 1998]. Let p be an arbitrarily large constant. There are two differences in the construction of G: (i) Q is a p-clique; and (ii) every v_i is attached by a loop of p vertices where each vertex is connected to Q by $k-2$ edges except v_i. Let $\gamma > 1-1/e$, if there is a solution with γ-approximation on optimal follower number for the edge k-core problem, there will be a λ-approximate solution on optimal element number for MC, where $\lambda = \gamma + \frac{(\gamma-1)\times b(d+3)}{p \times f}$ and f is the number of followers of edge k-core problem. Thus, the theorem is proved. The edge k-core problem is APX-hard. □

Theorem 3. *Let $f(A) = |\mathcal{F}(A)|$. We have that f is not submodular for $k \geq 2$.*

Algorithm 1 NaiveEKC

Input: G: a graph, k: degree constraint, b: budget
Output: a set A of anchor edges

1: $A \leftarrow \emptyset$; $C_k \leftarrow C_k(G)$
2: **for** i from 1 to b **do**
3: **for** each $e \in \binom{V}{2} \setminus \{E \cup A \cup \binom{V(C_k)}{2}\}$ **do**
4: compute $\mathcal{F}(e, G + A)$;
5: **end for**
6: $e^* \leftarrow$ the edge with most followers;
7: $A \leftarrow A \cup e^*$; $C_k \leftarrow C_k(G + A)$;
8: **end for**
9: **return** A

Proof. For two arbitrary collapsers sets A and B, if f is submodular, it must hold that $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$. Let Q_1 be a (k+1)-clique where vertices u and v are contained. Let Q_2 be another (k+1)-clique. We create a vertex w and connect it to the vertices in Q_2 by $k-2$ edges. If $A = (u, w)$ and $B = (v, w)$, $f(A) + f(B) = 0 < f(A \cup B) + f(A \cap B) = 1$. □

5 Solution

Due to the NP-hardness and inapproximability of the problem, we resort to a greedy heuristic which iteratively finds the best anchor, i.e., the edge with the largest number of followers. The framework of the greedy algorithm is shown in Algorithm 1. At Line 4, we compute the followers for each candidate anchor edge in the complement graph of G, except the anchored edges in A and the edges between the k-core vertices. Note that adding an edge between two (non-adjacent) k-core vertices cannot enlarge the k-core, because all the non-k-core vertices will still be deleted in k-core computation. After the computation for every candidate anchor edge, the best anchor is chosen at Line 6, and the k-core is updated by inserting the anchor edge. The time complexity of Algorithm 1 is $O(b \times n^2 \times m)$. As we do not explicitly record the candidate edges, the space complexity of Algorithm 1 is $O(m)$.

5.1 Optimizations on Each Iteration

In this section, we introduce optimization techniques for the first iteration of the greedy algorithm, i.e., the edge k-core problem with $b = 1$. They can be immediately applied to other iterations by replace the k-core of G by the edge k-core of $G + A$.

Basic Candidate Pruning

The following theorem locates the scope of valid candidate anchors where each edge has at least one follower.

Theorem 4. *Given a graph G, if a candidate edge $e = (u, v)$ has at least one follower, we have that $u \in C_{k-1}(G)$ and $v \in C_{k-1}(G)$, where at least one of u and v is in $H_{k-1}(G)$.*

Proof. Suppose $u \notin C_{k-1}(G)$. We have $e \in C_k(G_e)$; otherwise, we have $C_k(G_e) = C_k(G)$, i.e., there is no follower of e. If we remove e from $C_k(G_e)$, then $deg(u, C_k(G_e) \setminus \{e\})$ and $deg(v, C_k(G_e) \setminus \{e\})$ are at least $k-1$, because $(u, v) \in$

$C_k(G_e)$ and their degrees decrease by 1. Thus, $C_k(G_e) \setminus \{e\} \subseteq C_{k-1}(G)$ which contradicts with $u \notin C_{k-1}(G)$. Now we have proved that $u \in C_{k-1}(G)$ and $v \in C_{k-1}(G)$. Suppose that $\{u, v\} \in C_k(G)$ and $C_k(G_e) \setminus C_k(G) = F \neq \emptyset$, we have that $C_k(G_e) \setminus \{e\}$ belongs to $C_k(G)$, which contradicts with $F \notin C_k(G)$. □

Example 2. *In Figure 3, when $k = 3$, the 3-core $C_3(G)$ is induced by $\{v_0, v_1, v_5, v_6\}$, and the 2-shell $H_2(G) = \{v_2, v_3, v_7, v_8\}$. According to Theorem 4, we only need to consider every new edge (u, v) or (v, u) with $u \in H_2(G)$ and $v \in V(C_2(G)) = V(C_3(G)) \cup H_2(G)$, as candidates.*

Onion Layer based Candidate Pruning

Given the (k-1)-core of G, the computation of k-core on G recursively deletes some vertices in the (k-1)-core. The vertices are deleted in batch as their degrees are less than k at a same time, like peeling an onion [Zhang *et al.*, 2017c; Zhang *et al.*, 2017a]. The first layer of the onion, denoted by L_1, consists of the vertices in (k-1)-core with degree less than k, i.e., $L_1 = \{u \mid deg(u, C_{k-1}(G)) < k\}$. The deletion of the first layer vertices may decrease the degrees of other vertices, and produces the second layer. Recursively, we have that $L_i = \{u \mid deg(u, G_i) < k\}$ where $G_i = C_{k-1}(G) - G[\cup_{1 \leq j < i} L_j]$.

Let $l(u)$ denote the layer index of u, i.e., $u \in L_{l(u)}$. If $l(u) < l(v)$, we say u is at a lower layer of v, or v is at a higher layer of u. We use $\mathcal{L}$ to denote the union of layers, i.e., $\mathcal{L} = \cup_{1 \leq i \leq s} L_i$ where $s = max(\{l(u) \mid u \in H_{k-1}(G)\})$. Let $d^*(u)$ denote the number of adjacent neighbors of u in higher layers and k-core, i.e., $d^*(u) = deg(u, G')$ where $G' = C_{k-1}(G) - G[\{\cup_{1 \leq i \leq l(u)} L_i\} \setminus u]$.

Benefit from $\mathcal{L}$, we propose an effective pruning technique.

Theorem 5. *Given a graph G, if a candidate edge $e = (u, v)$ has at least one follower, we have that (i) $d^*(u) = k-1$ when $l(u) < l(v)$, (ii) $d^*(v) = k - 1$ when $l(v) < l(u)$, and (iii) $d^*(u) = d^*(v) = k - 1$ when $l(u) = l(v)$;*

Proof. Let O be the vertex deletion order of computing the k-core on the (k-1)-core. We have $d^*(u) \leq k - 1$ and $d^*(v) \leq k - 1$, since u and v are deleted at their layers in O, respectively. When $l(u) < l(v)$, we suppose $d^*(u) < k - 1$. The anchoring of (u, v) increases the degrees of u and v by 1, if we use the same order O to compute the k-core again, then $d^*(u) \leq k - 1 < k$, i.e., every vertex will be deleted at the same position of the order O, including u and v. It contradicts with $e = (u, v)$ has at least one follower. Thus, case (i) and (ii) are proved. When $l(u) = l(v)$, we suppose $d^*(u) < k-1$ without loss of generality. If we use the same order O to compute the k-core again with the anchored edge e, u will be deleted at the same position since $d^*(u) < k$. So v can't survive after the next batch of deletions since $d^*(v) < k$. Finally, every vertex will be deleted. □

Example 3. *In Figure 3, when $k = 3$, layer 1 contains v_3 and v_8, and layer 2 contains v_2 and v_7. By theorem 5, (v_2, v_7) is not a promising candidate edge because $d^*(v_7) = 1$, but (v_3, v_8) is a proper candidate edge.*

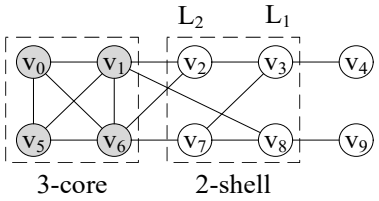


Figure 3: Candidate Seletion, $k = 3$

Note that if $l(u) \leq l(v)$, in the k-core computation with anchor e, the survive of u may preserve some vertices before visiting v. Thus, in Theorem 5, there is no degree requirement for v to ensure that $e = (u, v)$ has at least one follower.

Onion Layer based Follower Computation

A naive follower computation is to directly apply the k-core computation on the graph with the existence of an anchor. An improved idea is to use the core maintenance algorithms, which update the k-core for *every* k with the addition of an edge [Zhang *et al.*, 2017c]. For the edge k-core algorithm, we only need to update the k-core of a given k with an anchor edge. Thus, we adopt and refine the follower computation in the vertex-anchored k-core algorithm (OLAK) which is shown to be more efficient than core maintenance for k-core update with a fixed k [Zhang *et al.*, 2017a].

In OLAK, anchoring a vertex means the degree of the vertex is infinitely large. There is an observation that a vertex u is the follower of the anchor vertex x if there is a path $x \rightsquigarrow u$ based on neighboring relations, and $l(y) < l(z)$ for every two consecutive vertices y and z along the path. This indicates that we do not need to consider the vertices without such paths in the follower computation. Different from anchored k-core problem, edge k-core problem requires us to add a new edge, where the vertex degree is increased by exactly 1, and we have to consider two vertices rather than one.

In our algorithm, given an anchor edge (u, v) with $l(u) < l(v)$, we generate candidate followers layer-by-layer through activating the neighbors at higher layers, starting from the layer of u. Here u is the first activated vertex and only the activated vertices can activate their neighbors at higher layers. In this activation procedure, each activated vertex is assigned an upper bound of its degree in the edge k-core. The degree upper bound of a vertex u is generated by counting its neighbors in higher layers and the activated neighbors of u at other layers. Once the degree upper bound of a vertex is less than k, it will be deleted immediately and the upper bounds of its neighbours will be decreased by 1. The follower computation is complete when the layer-by-layer activation is finished. Note that the two vertices incident to the anchor edge may also be deleted in this procedure. Once one of them is deleted, there is no follower and the computation is returned.

Follower based Candidate Pruning

We can further prune some candidate edges by known results, when we retrieve the followers of some candidate edges.

Theorem 6. *Given a candidate anchor edge $e_1 = (u_1, v_1)$ and its follower set $\mathcal{F}(e_1)$, we have $\mathcal{F}(e) \subseteq \mathcal{F}(e_1)$ for every candidate edge $e \in \{(u, v) \mid u \in \mathcal{F}(e_1)$ and $v \in \mathcal{F}(e_1) \cup C_k(G)$ and $u \neq v\}$.*

Algorithm 2 EKC

Input: G: a graph, k: degree constraint, b: budget
Output: A set A of anchor edges

1: $A \leftarrow \emptyset$;
2: **for** i from 1 to b **do**
3: $N \leftarrow \{(u,v) \mid u \in H_{k-1}(G+A), v \in C_{k-1}(G+A)\} \setminus E(C_{k-1}(G+A))$ (Theorem 4);
4: compute $\mathcal{L}$ and filter N based on Theorem 5;
5: **for** each $e \in N$ **do**
6: $\mathcal{F}(e) \leftarrow$ FindFollower$(e, \mathcal{L})$ (Section 5.1);
7: update N based on Theorem 6;
8: **end for**
9: $e^* \leftarrow$ the edge with most followers; $A \leftarrow A \cup e^*$;
10: update N (Section 5.2);
11: **end for**
12: **return** A

Proof. Let $e = (u, v)$. If $\mathcal{F}(e)$ is not empty, then $C_k(G_e)$ contains $C_k(G)$, u, v and $\mathcal{F}(e)$. Because $C_k(G_{e_1})$ contains u and v, i.e., more vertices may be added to $C_k(G_e)$ after anchoring e_1. So, $V(C_k(G_e)) \subseteq V(C_k(G_{e_1}))$ and thus $\mathcal{F}(e) \subseteq \mathcal{F}(e_1)$. □

Example 4. *In Figure 3, when $k = 3$, if we get that v_2 and v_7 are the followers of edge (v_3, v_8). According to Theorem 6, the followers of (v_2, v_7) are also the followers of (v_3, v_8), thus (v_2, v_7) is not a promising candidate in current iteration.*

5.2 Reusing Intermediate Results across Iterations

When one iteration of the greedy algorithm is completed, we get the best anchor edge A and the number of followers for every candidate edge in this iteration. These results can be reused since some connected components in the induced subgraph of (k-1)-shell may keep the same topology after anchoring an edge. For each connected component S where none of the vertices are incident to the anchor A, we can record the largest number of followers of one candidate anchor in S, for the later iterations. For the connected component(s) S where there is a vertex incident to the anchor A, it is hard to reuse the results because the addition of edges may largely change the vertex deletion order and the layer structure in k-core computation.

5.3 EKC Algorithm

We present the EKC algorithm in Algorithm 2, which optimizes the greedy algorithm by adopting all the proposed techniques. At Line 3, the candidate edge set is restricted to N according to Theorem 4. In Line 4, we compute the onion layers of G and then exclude the candidates based on Theorem 5. The follower computation of a candidate edge is conducted by exploring $\mathcal{L}$ layer-by-layer, as introduced in Section 5.1. The set N is further filtered by Theorem 6 once a follower computation is completed. After each iteration, we get the anchor edge with the most followers. The algorithm terminates after b iterations.

The resulting set of b anchors is same to that in Algorithm 2. The correctness is guaranteed by the correctness of optimization techniques. The worst-case time complexity and space complexity are same to Algorithm 1.

Algorithm	Description
`Rand`	randomly chooses b anchor edges (each with at least one follower) from $N_1 = \{(u,v) \mid u \in H_{k-1}(G), v \in C_{k-1}(G)\} \setminus E(C_{k-1}(G))$
`Degree`	chooses b anchors from N_1 with largest degrees in $\mathcal{L}$
`Layer`	chooses b anchors from N_1 at highest onion layers in $\mathcal{L}$
`AKC`	the anchored k-core algorithm in [Zhang *et al.*, 2017a]
`Exact`	identifies the optimal solution by exhaustively searching all possible combinations of b anchors, with all the proposed techniques
`Naive`	computes a k-core on G for each candidate to find best anchor in each iteration (Algorithm 1)
`Baseline`	`Naive` + filtering candidates with Theorem 4
`BL+O1`	`Baseline` + filtering candidates with Theorem 5
`BL+O2`	`BL+O1` + filtering candidates with Theorem 6
`BL+OF`	`BL+O2` + $\mathcal{L}$ based follower computation (Section 5.1)
`EKC`	`BL+OF` + intermediate result reuse (Section 5.2)

Table 2: Summary of Algorithms

Dataset	Vertices	Edges	d_{avg}	k_{max}
`Facebook`	4,039	88,234	43.69	115
`Enron`	36,692	183,831	10.02	43
`Brightkite`	58,228	214,078	7.35	52
`Gowalla`	196,591	950,327	9.67	51
`DBLP`	317,080	1,049,866	6.62	113
`Twitter`	81,306	1,768,149	33.02	96
`Stanford`	281,903	2,312,497	14.14	71
`YouTube`	1,134,890	2,987,624	5.27	51
`Flickr`	513,969	3,190,452	12.41	309

Table 3: Statistics of Datasets

6 Evaluation

Algorithms. As far as we know, there is no existing algorithm for the edge k-core problem on general graphs. In the experiment, we implement and evaluate 11 algorithms as shown in Table 2, where the bottom 6 algorithms produce the same result, because they use the same greedy heuristic and candidate visiting order.

Datasets. `Flickr` is from http://networkrepository.com/, and the others are from https://snap.stanford.edu/data/. Table 3 shows the statistics of the datasets.

Settings. All programs are implemented in C++. All experiments are performed on Intel Xeon 2.20GHz CPU and Linux System. We vary the parameters k and b.

6.1 Effectiveness

Figure 4 reports the number of followers w.r.t. b anchors. For random approaches, we report the average number of followers for 100 independent tests. In Figure 4, `Degree` and `Layer` failed to get any followers in more than half of the datasets, because a large degree vertex or a vertex in higher

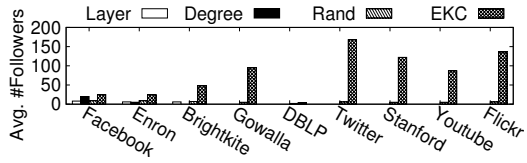


Figure 4: Effectiveness of the Greedy Heuristic, k=20, b=5

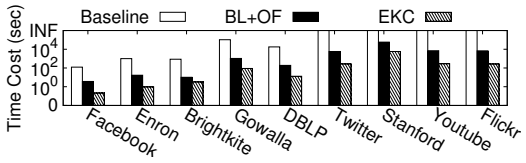
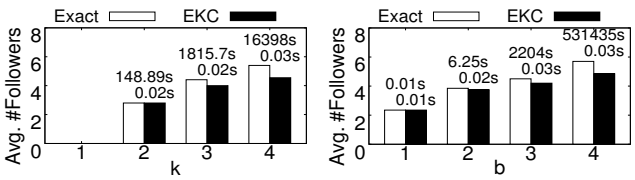
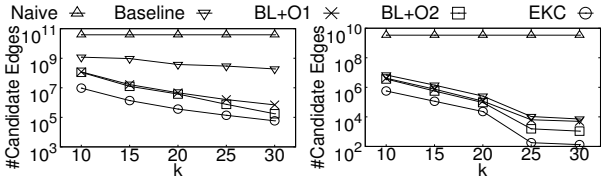


Figure 7: Running time on Different Datasets



(a) Gowalla, b=3 (b) Brightkite, k=3

Figure 5: Exact vs EKC



(a) Gowalla, b=20 (b) Brightkite, b=5

Figure 8: Candidate Pruning

onion layer does not necessarily have a follower. A totally random algorithm cannot get any followers in most settings. Thus, in `Rand`, we only choose the edges from the set where each edge has at least one follower. We observe that the majority of the candidate edges does not have any followers, while our greedy heuristic always finds effective anchors in the experiments. In Figure 4, we notice that `EKC` preserves more than 100 followers in `Flickr` with only 5 anchors.

We also compare the performance of `EKC` with the optimal solution from `Exact`. Due to extremely high cost, we run `Exact` on an induced subgraph by 50 random vertices. The results are from 100 independent settings. The values of k and b are small due to the small-scale of data. Figure 5 shows that the margins between `EKC` and `Exact` are not unacceptable, considering the non-submodular property and the significantly better efficiency of `EKC`.

Figure 6 shows that `EKC` produces similar and almost larger numbers of followers than `AKC` where the meaning of b is different: the former for edges and the latter for vertices. Although the anchoring of one vertex means the preserving of many edges, `EKC` can produce similar followers with one anchor edge. It shows that the edge k-core can enlarge the k-core with less graph manipulations.

6.2 Efficiency

Figure 7 reports the performance of three algorithms on all the datasets with $k = 20$ and $b = 5$. The datasets are ordered by the number of edges. We find the runtime among different datasets is largely influenced by the characteristics of datasets and the onion layer structures in k-core computation. `Baseline` cannot finish computation on 4 datasets within one week. `BL+OF` significantly improve the performance by applying a series of techniques. `EKC` performs even better given all the optimizations.

In Figure 8, we report the number of candidate edges of different algorithms by incrementally adding techniques, where the details of each algorithms is given in Table 3. We can see the improvement brought by each proposed technique.

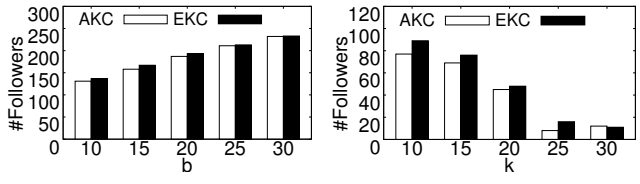
Figure 9 studies the impact of k and r. Figure 9(a) shows that the runtime decreases with a larger input of k. This is because the number of candidate edges become fewer with a larger k. Figure 9(b) reports the runtime with an increasing b, which is proportional to the value of b. The margin between `BL+OF` and `EKC` becomes smaller with a larger k, because the number of connected components becomes less when k increases. It is reported that `EKC` largely outperforms the other algorithms under all the settings.

7 Conclusion

In this paper, we investigate the problem of edge k-core, which aims to add a set b of edges in a network such that the size of the resulting k-core is maximized. We prove the problem is NP-hard and APX-hard. An efficient algorithm, named `EKC`, is proposed with novel optimizations. Extensive experiments on 9 real-life datasets are conducted to demonstrate our model is effective and our algorithm is efficient.

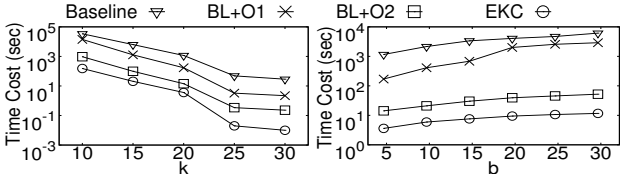
Acknowledgments

Xuemin Lin is supported by 2019DH0ZX01, 2018YFB1003504, NSFC61232006, ARC DP180103096 and DP170101628. Wenjie Zhang is supported by ARC DP180103096.



(a) Gowalla, k=20 (b) Brightkite, b=5

Figure 6: Anchor k-core vs Edge k-core



(a) Brightkite, b=5 (b) Brightkite, k=20

Figure 9: Runtime with Different k and b

References

[Batagelj and Zaversnik, 2003] Vladimir Batagelj and Matjaz Zaversnik. An o(m) algorithm for cores decomposition of networks. *CoRR*, cs.DS/0310049, 2003.

[Bhawalkar *et al.*, 2015] Kshipra Bhawalkar, Jon M. Kleinberg, Kevin Lewi, Tim Roughgarden, and Aneesh Sharma. Preventing unraveling in social networks: The anchored k-core problem. *SIAM J. Discrete Math.*, 29(3):1452–1475, 2015.

[Cheema *et al.*, 2010] Muhammad Aamir Cheema, Ljiljana Brankovic, Xuemin Lin, Wenjie Zhang, and Wei Wang. Multi-guarded safe zone: An effective technique to monitor moving circular range queries. In *ICDE*, pages 189–200, 2010.

[Chitnis and Talmon, 2018] Rajesh Chitnis and Nimrod Talmon. Can we create large k-cores by adding few edges? In *CSR*, pages 78–89, 2018.

[Chitnis *et al.*, 2013] Rajesh Hemant Chitnis, Fedor V. Fomin, and Petr A. Golovach. Preventing unraveling in social networks gets harder. In *AAAI*, 2013.

[Feige, 1998] Uriel Feige. A threshold of ln *n* for approximating set cover. *J. ACM*, 45(4):634–652, 1998.

[Kapron *et al.*, 2011] Bruce M. Kapron, Gautam Srivastava, and S. Venkatesh. Social network anonymization via edge addition. In *ASONAM*, pages 155–162, 2011.

[Karp, 1972] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972.

[Kitsak *et al.*, 2010] Maksim Kitsak, Lazaros K Gallos, Shlomo Havlin, Fredrik Liljeros, Lev Muchnik, H Eugene Stanley, and Hernán A Makse. Identification of influential spreaders in complex networks. *Nature physics*, 6(11):888–893, 2010.

[Lai *et al.*, 2005] Yung-Ling Lai, Chang-Sin Tian, and Ting-Chun Ko. Edge addition number of cartesian product of paths and cycles. *Electronic Notes in Discrete Mathematics*, 22:439–444, 2005.

[Luce and Perry, 1949] R Duncan Luce and Albert D Perry. A method of matrix analysis of group structure. *Psychometrika*, 14(2):95–116, 1949.

[Luo *et al.*, 2008] Yi Luo, Wei Wang, and Xuemin Lin. SPARK: A keyword search engine on relational databases. In *ICDE*, pages 1552–1555, 2008.

[Luo *et al.*, 2011] Yi Luo, Wei Wang, Xuemin Lin, Xiaofang Zhou, Jianmin Wang, and Keqiu Li. SPARK2: top-k keyword query in relational databases. *IEEE Trans. Knowl. Data Eng.*, 23(12):1763–1780, 2011.

[Malliaros and Vazirgiannis, 2013] Fragkiskos D. Malliaros and Michalis Vazirgiannis. To stay or not to stay: modeling engagement dynamics in social graphs. In *CIKM*, pages 469–478, 2013.

[Medya *et al.*, 2019] Sourav Medya, Tiyani Ma, Arlei Silva, and Ambuj K. Singh. K-core minimization: A game theoretic approach. *CoRR*, abs/1901.02166, 2019.

[Morone *et al.*, 2019] Flaviano Morone, Gino Del Ferraro, and Hernán A Makse. The k-core as a predictor of structural collapse in mutualistic ecosystems. *Nature Physics*, 15(1):95, 2019.

[Natanzon *et al.*, 2001] Assaf Natanzon, Ron Shamir, and Roded Sharan. Complexity classification of some edge modification problems. *Discrete Applied Mathematics*, 113(1):109–128, 2001.

[Seidman, 1983] Stephen B Seidman. Network structure and minimum degree. *Social networks*, 5(3):269–287, 1983.

[Suady and Najim, 2014] Suaad AA Suady and Alaa A Najim. On edge-addition problem. *Journal of College of Education for Pure Science*, 4(1):26–36, 2014.

[Ugander *et al.*, 2012] Johan Ugander, Lars Backstrom, Cameron Marlow, and Jon Kleinberg. Structural diversity in social contagion. *PNAS*, 109(16):5962–5966, 2012.

[Wu *et al.*, 2013] Shaomei Wu, Atish Das Sarma, Alex Fabrikant, Silvio Lattanzi, and Andrew Tomkins. Arrival and departure dynamics in social networks. In *WSDM*, pages 233–242, 2013.

[Zhang *et al.*, 2017a] Fan Zhang, Wenjie Zhang, Ying Zhang, Lu Qin, and Xuemin Lin. OLAK: an efficient algorithm to prevent unraveling in social networks. *PVLDB*, 10(6):649–660, 2017.

[Zhang *et al.*, 2017b] Fan Zhang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. Finding critical users for social network engagement: The collapsed k-core problem. In *AAAI*, pages 245–251, 2017.

[Zhang *et al.*, 2017c] Yikai Zhang, Jeffrey Xu Yu, Ying Zhang, and Lu Qin. A fast order-based approach for core maintenance. In *ICDE*, pages 337–348, 2017.

[Zhang *et al.*, 2018a] Fan Zhang, Long Yuan, Ying Zhang, Lu Qin, Xuemin Lin, and Alexander Zhou. Discovering strong communities with user engagement and tie strength. In *DASFAA*, pages 425–441, 2018.

[Zhang *et al.*, 2018b] Fan Zhang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. Efficiently reinforcing social networks over user engagement and tie strength. In *ICDE*, pages 557–568, 2018.

[Zhu *et al.*, 2018] Weijie Zhu, Chen Chen, Xiaoyang Wang, and Xuemin Lin. K-core minimization: An edge manipulation approach. In *CIKM*, pages 1667–1670, 2018.