Finding Critical Users for Social Network Engagement: The Collapsed k-Core Problem

Fan Zhang,[†] Ying Zhang,[†] Lu Qin,[†] Wenjie Zhang,[§] Xuemin Lin[§]
[†]QCIS, University of Technology Sydney, [§]University of New South Wales
fanzhang.cs@gmail.com, {ying.zhang, lu.qin}@uts.edu.au, {zhangw, lxue}@cse.unsw.edu.au

Abstract

In social networks, the leave of critical users may significantly break network engagement, i.e., lead a large number of other users to drop out. A popular model to measure social network engagement is k-core, the maximal induced subgraph in which every vertex has at least k neighbors. To identify critical users for social network engagement, we propose the collapsed k-core problem: given a graph G, a positive integer k and a budget b, we aim to find b vertices in G such that the deletion of the b vertices leads to the smallest k-core. We prove the problem is NP-hard. Then, an efficient algorithm is proposed, which significantly reduces the number of candidate vertices to speed up the computation. Our comprehensive experiments on 9 real-life social networks demonstrate the effectiveness and efficiency of our proposed method.

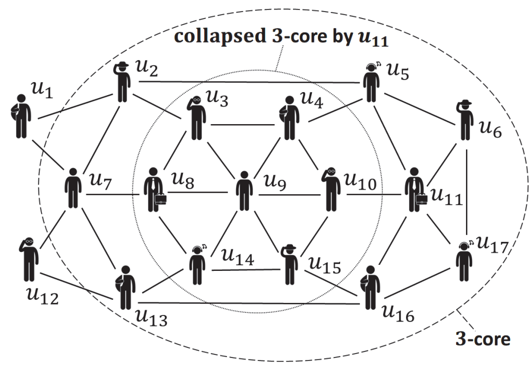


Figure 1: Motivating Example

Introduction

The user engagement on social network has attracted significant interests over recent years (Wang et al. 2016; Wu et al. 2013; Bhawalkar et al. 2015). k-core is a simple and popular model based on degree constraint, which has been widely used to measure the network engagement (Malliaros and Vazirgiannis 2013; Chitnis, Fomin, and Golovach 2013; 2016; Abello and Queyroi 2013; Garcia, Mavrodiev, and Schweitzer 2013). Assuming all users in a community/group are initially engaged, each individual has two strategies, to remain engaged or drop out. Particularly, a user will remain engaged if and only if at least k of his/her friends are engaged (i.e., degree constraint). A user with less than k friends engaged will drop out, and his/her leave may be contagious and forms a cascade of the departure (i.e., collapse) in the network. When the collapse stops, the remaining engaged users corresponds to the well-known concept k-core, the maximal induced subgraph in which every vertex has at least k neighbors. The size of k-core can be used to measure the overall engagement of the social network.

A natural question is that, given a limited budget b, how to find b vertices (i.e., users) in a network so that we can get the smallest k-core by removing these b vertices. This problem is named the collapsed k-core problem in this paper, which aims to collapse the engagement of the network with the greatest extent for a given budget b. By developing an efficient and scalable solution for this problem, we can quickly identify critical users whose leave will collapse the network most severely. These users are critical for the overall engagement of social networks. For instance, we can find most valuable users, to sustain or destroy the engagement of the networks. We can also evaluate the robustness of network engagement against the vertex attack.

Example 1. *Suppose there is a study group, and the number of friends in the group reflects the willingness of engagement for each member (i.e., user). If one drops out, he/she will weaken the willingness of his/her friends to remain engaged, which may incur the collapse of the group. As illustrated in Figure 1, we model* 17 *members in a study group and their relationship as a network. According to the above engagement model with k=3, i.e., a person will drop out if there are less than* 3 *friends,* 15 *members will remain engaged; that is,* 3*-core of the network is the whole network excluding* u_1 *and* u_{12}*. Clearly, if users in* 3*-core drop out regardless the number of friends, e.g., attracted by another group, the network will further collapse. The extent of the collapse varies among different users. For instance, although* u_9 *has* 6 *friends in* 3*-core, the departure of* u_9 *will not further lead to the leave of other users because each of his/her neighbors still has* 3 *friends engaged. On the contrary, the leave of* u_{11} *will lead to the leave of 7 members in the group including* u_2*,* u_5*,* u_6*,* u_7*,* u_{13}*,* u_{16}*, and* u_{17}*. In this sense, it is more cost-effective to give* u_{11} *the incentive (e.g., bonus) to ensure his/her en-*

Notation	Definition
G	an unweighted and undirected graph
u, v, x	vertex in the graph
n, m	the number of vertices and edges in G
A	a set of collapsers vertices
G_x (G_A)	graph G collapsed by x (A)
k	the degree constraint
b	the budget for the number of collapsers
$C_k(G)$	k-core of G
$\|C_k\|$	the number of vertices in $C_k(G)$
$C_k(G_A)$	collapsed k-core with vertices in A deleted
$\mathcal{F}(x)$ ($\mathcal{F}(A)$)	followers of a collapser x (A)
$deg(u, S)$	the number of adjacent vertices of u in S
$NB(u, S)$	the adjacent vertices of u in S

Table 1: Summary of Notations

gagement or persuade him/her to leave the group.

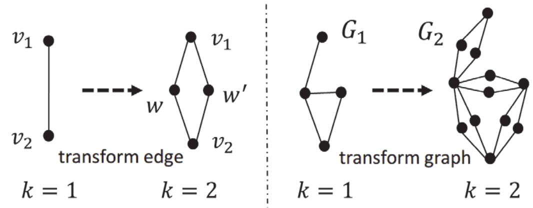
Challenges and Contributions. To the best of our knowledge, we are the first to propose and investigate the collapsed k-core problem. We prove the problem is NP-hard for any k value. To avoid enumerating all possible answer sets with size b, we resort to greedy heuristics where the best vertex is obtained in each iteration. Through theoretical analyses, we significantly reduce the number of candidate vertices to speed up the computation. We develop an efficient algorithm, namely CKC, to solve the collapsed k-core problem. Our comprehensive experiments on 9 real-life networks demonstrate the efficiency of our proposed techniques. A real example from `DBLP` is presented in the experiments to show the effectiveness of the collapsed k-core model.

Preliminaries

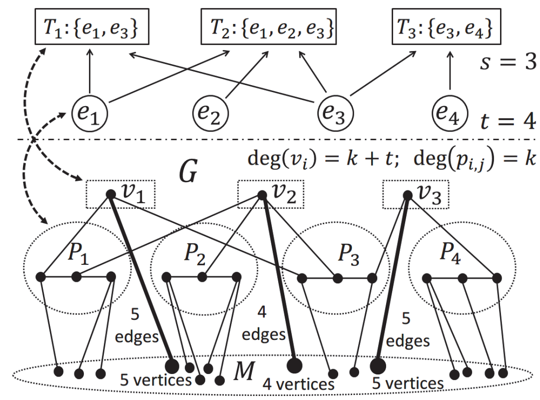
We consider an unweighted and undirected graph $G = (V, E)$, where V (resp. E) represents the set of vertices (resp. edges) in G. When the context is clear, we use a set S of vertices to represent the induced subgraph of G with $S \subseteq G$. We use n (resp. m) to denote the number of vertices (resp. edges) in the graph G and we assume $m > n$. We denote the adjacent vertices set of u in G by $NB(u, G)$, which is also called the neighbors set of u in G. We use $deg(u, S)$, the degree of u in S, to represent the number of adjacent vertices of u in S. $NB(u, S)$ (resp. $deg(u, S)$) is also written as $NB(u)$ (resp. $deg(u)$) when the context is clear. Given a subgraph S, $NB(S)$ denotes the union of the neighbors of the vertices in S. The concept of k-core has been widely used to describe cohesive subgraphs, which is formally defined as follows.

Definition 1. ***k-core**. Given a graph G and a positive integer k, an induced subgraph S is the k-core of G, denoted by $C_k(G)$, if (i) S satisfies degree constraint, i.e., $deg(u, S) \geq k$ for every $u \in S$; and (ii) S is maximal, i.e., any subgraph $S' \supset S$ cannot be a k-core.*

The k-core of a graph G can be obtained by recursively removing the vertices whose degrees are less than k, with time complexity $\mathcal{O}(m)$ (Batagelj and Zaversnik 2003). In real applications, the value of k is determined by users based on their requirement for cohesiveness. The resulting k-core will be more cohesive if the k value becomes larger.



(a) Construction Example, $k = 2$



(b) Construction Example, $k = 3$ (can extend to $k > 3$)

Figure 2: Examples for NP-hardness Proof

In this paper, once a vertex u in G is **collapsed**, it is always removed from k-core regardless of the degree constraint.

Definition 2. ***collapsed** k**-core**. Given a graph G and a set $A \subseteq G$ of vertices, the collapsed k-core, denoted by $C_k(G_A)$, is the corresponding k-core of G with vertices in A removed.*

In addition to the deletion of the collapsed vertices in A, more vertices in $C_k(G)$ might be deleted as well due to the contagious nature of the k-core computation. These vertices are called **followers** of the collapsed vertices A, denoted by $\mathcal{F}(A, G)$, because they will remain in k-core if the vertices in A are not deleted. The size of the followers reflects the effectiveness of the collapsed vertices, where $\mathcal{F}(A, G) = C_k(G) \setminus \{C_k(G_A) \cup A\}$. In the following, we may use **collapsers** to represent the collapsed vertices, and use $\mathcal{F}(A)$ to denote $\mathcal{F}(A, G)$ when the context is clear.

Problem Statement. Given a graph G, a degree constraint k and a budget b, the **collapsed k-core problem** aims to find a set A of b collapsed vertices in G so that the size of the resulting collapsed k-core, $C_k(G_A)$, is minimized; that is, $\mathcal{F}(A, G)$ is maximized.

Example 2. *In Figure 1, if we set $k = 3$ and $b = 1$, the result of the collapsed k-core problem can be $A = \{u_{11}\}$ with $C_k(G_A) = \{u_3, u_4, u_8, u_9, u_{10}, u_{14}, u_{15}\}$ and $\mathcal{F}(A, G) = \{u_2, u_5, u_6, u_7, u_{13}, u_{16}, u_{17}\}$.*

Complexity

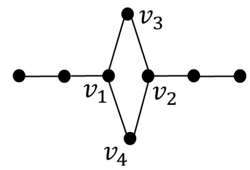
Theorem 1. *The collapsed k-core problem is NP-hard for any k.*

Proof. (1) When $k = 1$, we reduce the collapsed k-core problem to the maximum independent set problem (Woeginger 2001). To delete a vertex from 1-core during the collapsed 1-core computation, we have to remove all its adjacent vertices, i.e., make the vertex independent. Consequently, the problem of finding the maximum independent set S in a graph G is equivalent to finding the set of vertices $G \setminus S$ such that $G \setminus S$ is minimum and collapsing them can lead to an empty 1-core. Note that we need to try at most $n-1$ times ($1 \leq b < n$) to find the minimum $G \setminus S$. Thus, we have the collapsed k-core problem is NP-hard when $k = 1$.

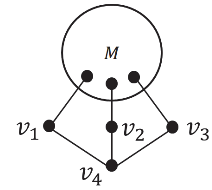
(2) When $k = 2$, we reduce the collapsed 2-core problem to the case of $k = 1$, which has been proved to be NP-hard. Given any graph G_1 with n vertices and m edges, we construct another graph G_2 with $n + 2m$ vertices and $4m$ edges as follows. For each edge (v_1, v_2) in G_1, we add two virtual vertices w and w' and construct the following four edges in G_2: (v_1, w), (w, v_2), (v_1, w') and (w', v_2), as shown in Figure 2 (a). An example of graph construction is also illustrated in Figure 2 (a). We do not need to include any virtual vertices in the optimal solution of collapsed 2-core because the influence of deleting a virtual vertex can always be covered by deleting one of its two neighbor vertices (non-virtual vertices). Therefore, the deletion of each edge in G_1 during the computation is always mapped to the deletion of four corresponding edges in G_2. Then the optimal solution of collapsed 2-core on G_2 is also that of collapsed 1-core on G_1. As a result, the collapsed k-core problem is NP-hard when $k = 2$.

(3) When $k \geq 3$, we reduce the collapsed k-core problem to the maximum coverage problem (Karp 1972); that is finding at most b sets to cover the largest number of elements, where b is a given budget. Firstly, we consider an arbitrary instance of maximum coverage problem with s sets $T_1, .., T_s$ and t elements $\{e_1, .., e_t\} = \cup_{1 \leq i \leq s} T_i$. Then we construct a corresponding instance of the collapsed k-core problem in a graph G as follows.

The set of vertices in G consists of three parts: M, V, and P. M consists of $(t+s)^4$ vertices in which every pair of vertices in M are adjacent. V consists of s vertices, v_1, $v_2, \ldots, v_s$, where vertex v_i corresponds to the set T_i for any $1 \leq i \leq s$. For each vertex v_i ($1 \leq i \leq s$), we add $k+t-|T_i|$ edges from v_i to $k + t - |T_i|$ unique vertices in M. Here, by unique, we mean that each vertex in M can be used at most once when adding edges to vertices outside M. P consists of t parts $P_1, P_2, \ldots, P_t$, where each part P_i ($1 \leq i \leq t$) corresponds to the element e_i and P_i consists of s vertices $p_{i,1}$,$p_{i,2}, \ldots, p_{i,s}$. For each P_i ($1 \leq i \leq t$) we first add $s-1$ edges, that is, for each $1 \leq j < s$, we add an edge from $p_{i,j}$ to $p_{i,j+1}$. For each set T_i ($1 \leq i \leq s$) and each element e_j ($1 \leq j \leq t$), if $e_j \in T_i$, we add an edge $(v_i, p_{j,i})$ in G. At this stage, the degree of each vertex in P is at most 3. Next, we add edges from vertices in P to unique vertices in M to guarantee that the degree of each vertex in P is exactly k. This can be done since $k \geq 3$. Then the construction of G is completed. Clearly, G is a k-core. Figure 2 (b) shows an example of the graph G with $k = 3$ constructed from 3 sets and 4 elements.



(a) $k = 1$



(b) $k = 2$ (can extend to $k > 2$)

Figure 3: Examples for Non-submodular

The key idea is that we ensure that: (i) only vertices in V need to be considered as collapsed vertices, since any vertex in M or P cannot have more followers than a vertex in V; (ii) none of the vertices in M will be deleted during the computation; (iii) all P_i have the same size for $1 \leq i \leq t$; and (iv) when a vertex v_i ($1 \leq i \leq s$) is removed, for each part P_j ($1 \leq j \leq t$) connected with v_i (i.e., $e_j \in T_i$), all vertices in P_j will be deleted due to degree constraint. By doing this, the optimal solution of the collapsed k-core problem corresponds to optimal solution of the maximum coverage problem. Since the maximum coverage problem is NP-hard, we prove that the collapsed k-core problem is NP-hard for any $k \geq 3$. □

We also show the properties of monotone and non-submodular towards the collapsed k-core problem in Theorem 2.

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

Proof. Suppose there is a set $A' \supseteq A$. For every vertex u in $\mathcal{F}(A)$, u will still be deleted in the collapsed k-core with the collapsers set A', because removing vertices in $A' \setminus A$ cannot increase the degree of u. Thus $f(A') \geq f(A)$ and f is monotone. For two arbitrary collapsers sets A and B, if f is submodular, it must hold that $f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$. We show that the inequality does not hold using counterexamples. When $k = 1$, we use the example shown in Figure 3 (a). Suppose $k = 1$, $A = \{v_1\}$ and $B = \{v_2\}$, we have $\mathcal{F}(A \cup B) = \{v_3, v_4\}$, $\mathcal{F}(A \cap B) = \mathcal{F}(A) = \mathcal{F}(B) = \emptyset$, so the inequation does not hold. When $k = 2$, we use the example shown in Figure 3 (b). Here, M is a complete graph with $4 \times k$ vertices. When $k = 2$, if $A = \{v_1\}$ and $B = \{v_2\}$, we have $\mathcal{F}(A \cup B) = \{v_3, v_4\}$, $\mathcal{F}(A \cap B) = \mathcal{F}(A) = \mathcal{F}(B) = \emptyset$, so the inequation does not hold. When $k > 2$, we add $k-2$ edges between v_i and M, for each $1 \leq i \leq 4$. We can prove that for $A = \{v_1\}$ and $B = \{v_2\}$, the inequation is still violated. □

Solution

Motivation

A straightforward solution of the collapsed k-core problem is to exhaustively enumerate all possible set A with size b, and compute the resulting collapsed k-core for each possible A. The time complexity of $\mathcal{O}(\binom{n}{b}m)$ is cost-prohibitive. Considering the NP-hardness of the problem, we resort to

Algorithm 1: **GreedyCKC(G, k, b)**

```
  Input  : G : a social network, k : degree constraint,
           b : number of collapsers
  Output : A : the set of collapsers
1 A := ∅; i := 0;
2 while i < b do
3   for each u ∈ C_k(G_A) do
4     Compute F(A ∪ u, G);
5   u* ← the best collapser in this iteration;
6   A := A ∪ u*; i := i + 1; update C_k(G_A);
7 return A
```

Algorithm 2: **CKC(G, k)**

```
  Input  : G : a social network, k : degree constraint,
  Output : x : the best collapser
1 C_k(G) := compute C_k(G);
2 P := {u : deg(u, C_k(G)) = k};
3 T := P ∪ {u : u ∈ C_k(G) & NB(u, G) ∩ P ≠ ∅};
4 for each u ∈ T (Theorem 3) do
5   Compute F(u, G);
6   T := T \ F(u, G) (Theorem 4);
7 return the best collapser
```

the greedy heuristic which iteratively finds the best collapser, i.e., the vertex with the largest number of followers. Clearly, we only need to consider the vertices in $C_k(G_A)$ since all other vertices will be deleted by degree constraint during k-core computation. Thus, a greedy algorithm is shown in Algorithm 1 with time complexity $\mathcal{O}(bnm)$, where n and m correspond to the number of candidate collapsers in each iteration (Line 3) and the cost of follower computation (Line 4), i.e., k-core computation.

The number of vertices in $C_k(G_A)$ at Line 3 is still considerably large, which motivates us to develop two effective pruning rules to further reduce the candidate vertices in each iteration of the greedy algorithm. Details are introduced in the following subsection.

Reducing Candidate Collapsers

For presentation simplicity, in this subsection, we introduce two pruning rules to find the vertex with the largest number of followers in the first iteration of the greedy algorithm (i.e., $A = \emptyset$). They can be immediately extended to the following iterations of the greedy algorithm by using the updated $C_k(G_A)$ to replace $C_k(G)$.

Theorem 3 indicates that only vertices with degree k in k-core and their neighbors in k-core can have followers. Particularly, P denotes the vertices in k-core of G with degree k, while T represents vertices in P as well as their neighbors within k-core.

Theorem 3. *Given a graph G and the set $P = \{u : deg(u, C_k(G)) = k\}$, if a collapsed vertex x has at least one follower, x is from T where $T = P \cup \{u : u \in C_k(G)$ & $NB(u,G) \cap P \neq \emptyset\}$; that is $|\mathcal{F}(x,G)| > 0$ implies $x \in T$.*

Proof. We prove that a vertex $x \in G \setminus T$ cannot have any follower. (1) If $x \in G \setminus C_k(G)$, x will be deleted in k-core computation and hence $|\mathcal{F}(x)| = 0$. (2) If $x \in C_k(G) \setminus T$, x survived in k-core computation and for each x's neighbor u within $C_k(G)$, we have $deg(u, C_k(G)) > k$ since $x \notin T$. Consequently, if x is deleted, we have $deg(u, C_k(G)) \geq k$; that is, the removal of x cannot be propagated to any of its neighbors regarding degree constraint and hence other vertices. It means x does not have any follower. Since $(G \setminus C_k(G)) \cup$ $(C_k(G) \setminus T)$ $\cup T = G$, we have $|\mathcal{F}(x,G)| > 0$ implies $x \in T$. □

In the following theorem, we further reduce the candidate vertices by excluding vertices which have been identified as followers of other vertices.

Theorem 4. *Given two vertices x and u in graph G, we have $\mathcal{F}(u) \subset \mathcal{F}(x)$ if $u \in \mathcal{F}(x)$.*

Proof. $u \in \mathcal{F}(x)$ implies that u will be deleted if x is collapsed. For every vertex in $\mathcal{F}(u)$, if x is collapsed, it will also be deleted since u will be deleted and collapsing x cannot increase degrees for vertices. Thus $\mathcal{F}(u) \subseteq \mathcal{F}(x)$. Since $u \in \mathcal{F}(x)$ and $u \notin \mathcal{F}(u)$, we have $\mathcal{F}(u) \subset \mathcal{F}(x)$. □

According to Theorem 4, in the procedure of finding a best collapser, every vertex which is a follower of a vertex can be excluded from candidate collapsers. Consequently, checking promising collapsers first, which may have large number of followers, can skip more vertices in the computation. Naturally, a vertex with more neighbors in the set P is more promising because all its neighbors in P will follow the vertex to be deleted. Thus, to further reduce the number of candidate collapsers, we try collapsing vertices in decreasing order of their degrees in P.

CKC Algorithm

By taking advantage of two pruning rules in Theorems 3 and 4, Algorithm 2 illustrates the details of CKC algorithm which finds the best collapser for a given graph G (i.e., $b = 1$). Particularly, we first compute the k-core of graph G (Line 1) and find the set P of vertices with degree k in C_k (Line 2). According to Theorem 3, we find the set T of vertices in P, and vertices which are inside C_k and are neighbors of at least one vertex in P (Line 3). To compute $\mathcal{F}(u, G)$, we can continue the k-core computation in Line 1 with vertex u deleted (Line 5). We have the best collapser when the algorithm terminates.

To handle the general case with $b > 1$, our CKC algorithm can be easily fit to the greedy algorithm (replacing Line 3 and 4) to find the best collapser in each iteration. In order to avoid the re-computation of P (Line 2) and T (Line 3) in the following iterations, we incrementally update two sets at the end of each iteration. Specifically, let P_1 denote the vertices whose degrees are decreased to k during the computation and P_2 denote the vertices which are discarded during the computation, we have $P = P \cup (P_1 \setminus P_2)$; Towards the set T, we include new vertices in $NB(P_1)$ and delete vertices in $NB(P_2)$ which do not have any neighbor in the updated P.

Additionally, if we find a vertex $u \in F(x)$ in one iteration of Algorithm 1, x is always a better candidate collapser than u

Dataset	Vertices	Edges	d_{avg}	$\lvert C_{20} \rvert$
`Facebook`	4,039	88,234	43.7	1,854
`Brightkite`	58,228	194,090	6.7	900
`Gowalla`	196,591	456,830	4.7	3,841
`Yelp`	552,339	1,781,908	6.5	20,839
`YouTube`	1,134,890	2,987,624	5.3	18,890
`DBLP`	1,566,919	6,461,300	8.3	29,564
`Pokec`	1,632,803	8,320,605	10.2	10,817
`LiveJournal`	3,997,962	34,681,189	17.4	469,951
`Orkut`	3,072,441	117,185,083	76.3	2,242,775

Table 2: Statistics of Datasets

in following iterations, because deleting other vertices cannot change the fact that x has more followers than u (Theorem 4). Actually, we do not need to consider u as a candidate in following iterations because u will be excluded from k-core whenever x is removed. In our implementation, we order the candidates by their number of neighbors in P in each iteration to prune more candidate collapsers.

Evaluation

This section evaluates the effectiveness and efficiency of the proposed techniques through comprehensive experiments.

Experimental Setting

Algorithms To the best of our knowledge, there is no existing work investigating the collapsed k-core problem and corresponding algorithms. In this paper, we implement and evaluate the following algorithms.

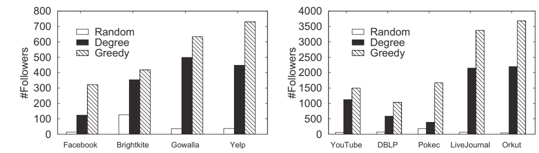
- $\mathbf{Baseline}$. The baseline greedy algorithm (Algorithm 1). In each iteration, it conducts collapsed k-core computation on every vertex in the updated k-core to find the best collapser.
- $\mathbf{CKC}$. The greedy algorithm in which collapsed k-core algorithm (Algorithm 2) is used in each iteration.

Datasets 9 real-life networks are deployed in our experiments and we assume all vertices in each network are initially engaged. The original data of `Yelp` is from https://www.yelp.com.au/dataset_challenge, `DBLP` is from http://dblp.uni-trier.de/ and the others are from http://snap.stanford.edu/. Table 2 shows statistics of 9 datasets which are listed in increasing order of their edge numbers.

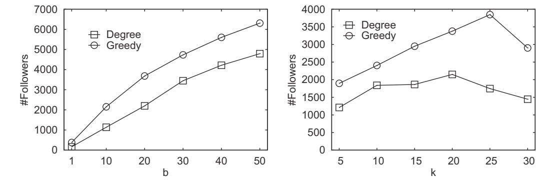
All programs are implemented in standard C++ and compiled with G++ in Linux. All experiments are performed on a machine with Intel Xeon 2.8GHz CPU and Redhat Linux System. We evaluate the effectiveness of the algorithms by reporting the number of the followers for resulting collapsers. The efficiency of the algorithms is measured by running time and the number of vertices accessed.

Effectiveness

We compare the number of followers produced by CKC with the results of other approaches, and also conduct a case study to demonstrate a detailed example of the collapsed k-core.

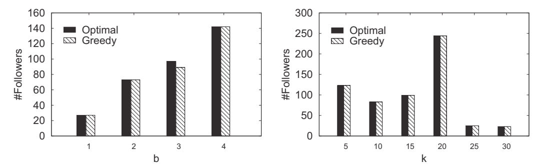


(a) 4 Datasets, k=20, b=20 (b) 5 Datasets, k=20, b=20



(c) Orkut, k=20 (d) LiveJournal, b=20

Figure 4: Number of the Followers



(a) Facebook, k=20 (b) Brightkite, b=2

Figure 5: Greedy vs Optimal

Effectiveness of the Greedy Algorithm Figure. 4 compares the number of followers w.r.t b collapsers identified by CKC algorithm with that of two other approaches, in which one randomly chooses b collapsers from vertices in k-core (**Random**) and the other chooses b collapsers in the candidate set T (Theorem 3) with the largest degrees (**Degree**). For $Random$, we report the average number of the followers for 100 independent testings . Figure. 4 (a) and (b) show that although $Degree$ based approach significantly improves the performance, but it is outperformed by our approach with a big margin. This implies that it is not effective to find collapsers simply based on degree information. Figure 4 (c) and (d) report the impact of b and k on the number of followers for CKC algorithm. The number of the followers clearly grows with the increase of budget b. The number becomes relatively small when k is small or large.

To further justify the effectiveness of the greedy approach, we also compare its performance with that of optimal algorithm (**Optimal**), which conducts exhaustively search on two relatively small networks with b varying from 1 to 4 on `Facebook` and k varying from 5 to 30 on `Brightkite`. Figure 5 shows that the greedy algorithm achieves the optimal solution except under one setting.

Case Study on DBLP Figure 6 depicts the collapser identified by the greedy algorithm on `DBLP` with $b = 1$ and $k = 20$ as well as the corresponding followers. For a clear presentation, edges between each author and authors in k-core are integrated as one edge. It is interesting that the author "Ying

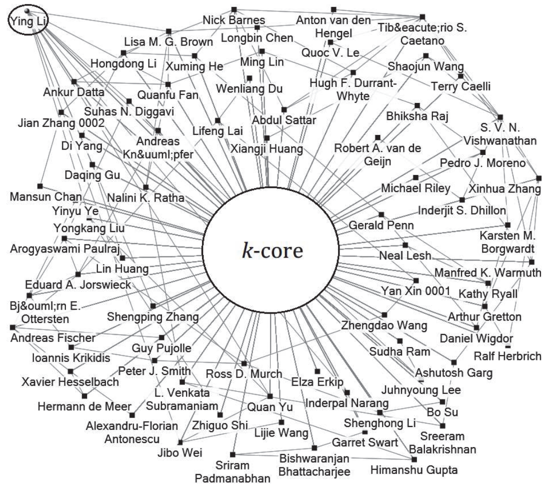
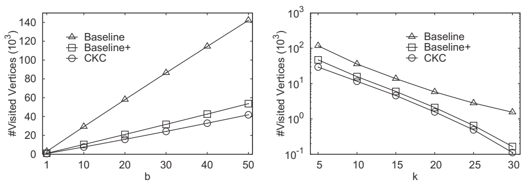


Figure 6: Case Study on DBLP, k=20, b=1



(a) DBLP, k=20 (b) DBLP, b=20

Figure 7: Effectiveness of Reducing Candidate Collapsers

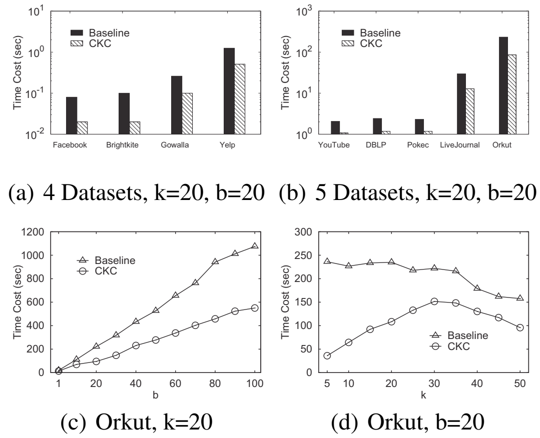
Li" alone has 74 followers, and only 12 of them are neighbors of "Ying Li". Moreover, we observe that the followers includes many professors and at least one IEEE fellow (Nalini K. Ratha). This shows the overall engagement of the network can be severely damaged by the leave of a few individuals.

Efficiency

We first investigate the efficiency of the individual techniques, then compare our CKC algorithm with Baseline.

Evaluation of Individual Techniques Figure 7 reports the number of visited vertices, i.e., the size of candidate collapsers, in three algorithms. Algorithm **Baseline+** represents $Baseline$ algorithm equipped with candidate collapsers reducing technique (Theorem 3). We can see the number of visited vertices significantly drops by Theorem 3 on `DBLP` for different k and b. It is reported that Theorem 4 further reduces the number of candidate collapsers, which is used in algorithm CKC.

Performance Evaluation Figures 8 (a) and (b) report the performance of two algorithms on 9 networks with $k = 20$ and $b = 20$. Datasets are ordered by their network sizes (i.e., the number of edges) where the largest network `Orkut` has 117 million edges. We can see CKC runs several times faster than $Baseline$ on all datasets. It is shown that CKC is also scalable to the growth of the network size, which identifies a set of 20 collapsers in 110 seconds on `Orkut`. Figures 8 (c)



(a) 4 Datasets, k=20, b=20 (b) 5 Datasets, k=20, b=20

(c) Orkut, k=20 (d) Orkut, b=20

Figure 8: Performance of the Algorithms

and (d) study the impact of k and b on two algorithms against `Orkut`, with b varying from 1 to 100 and k ranging from 5 to 50. We can see CKC is scalable towards the growth of b and outstanding on running time for different k, especially for small or large k. It is reported that CKC significantly outperforms $Baseline$ under all settings.

Related Work

k-core computation is first introduced by Seidman (Seidman 1983) and becomes a fundamental graph problem with a wide spectrum of applications such as social contagion (Ugander et al. 2012), network analysis (Adiga and Vullikanti 2013), network visualization (Zhang and Parthasarathy 2012; Zhao and Tung 2012), event detection (Meladianos et al. 2015), internet topology (Carmi et al. 2007), dense subgraph problems (Andersen and Chellapilla 2009), influence study (Kitsak et al. 2010; Vogiatzis 2013), graph clustering (Giatsidis et al. 2014), graph model validation (Healy et al. 2006), structure analysis of software system (Zhang et al. 2010), and protein function prediction (Altaf-Ul-Amine et al. 2003). There are multiple studies for core number computation under different settings including a linear-time in-memory algorithm (Batagelj and Zaversnik 2003), I/O efficient algorithms (Wen et al. 2016; Cheng et al. 2011), locally computing and estimating (Cui et al. 2014) and core number maintenance on dynamic graphs (Aksu et al. 2014; Zhang et al. 2016).

The engagement dynamic in social networks has attracted significant focus, e.g., (Wang et al. 2016; Chwe 2000; Bhawalkar et al. 2015; Malliaros and Vazirgiannis 2013; Wu et al. 2013; Chitnis, Fomin, and Golovach 2013; 2016). The k-core becomes more and more popular in social studies, because its degeneration property can be used to quantify engagement dynamics in real social networks (Malliaros and Vazirgiannis 2013). In the problem of anchoring b vertices to increase the k-core size (Bhawalkar et al. 2015), we need to consider the vertices *not* in k-core because it is useless to anchor vertices already in k-core. This is different from the problem of collapsed k-core, where the deletion of a vertex *not* in k-core will not affect the resulting k-core. As mentioned in the proof of NP-hardness, the independent set and the maximum coverage problems can match certain cases of the collapsed k-core problem, thus their solutions may be helpful in solving special cases of our problem, while they cannot be applied to solving the complete problem. To the best of our knowledge, our paper is the first to study the collapsed k-core problem to find critical users for social network engagement.

Conclusion

In this paper, we propose and study the problem of collapsed k-core, which intends to find a set of vertices whose deletion can lead to the smallest k-core of the network. We prove the problem is NP-hard for any given k. An efficient algorithm is proposed, which significantly reduces the number of candidate vertices to speed up the computation. Empirical study shows our method can find critical users in the network whose leave leads a large number of users to drop out. Extensive experiments on 9 real-life networks demonstrate our method is scalable on large size networks.

Acknowledgments

Ying Zhang is supported by ARC DE140100679 and DP170103710. Lu Qin is supported by ARC DE140100999 and DP160101513. Wenjie Zhang is supported by ARC DP150103071 and DP150102728. Xuemin Lin is supported by NSFC61232006, ARC DP150102728, DP140103578 and DP170101628.

References

Abello, J., and Queyroi, F. 2013. Fixed points of graph peeling. In *ASONAM*, 256–263.

Adiga, A., and Vullikanti, A. K. S. 2013. How robust is the core of a network? In *ECML-PKDD*, 541–556.

Aksu, H.; Canim, M.; Chang, Y.-C.; Korpeoglu, I.; and Ulusoy, Ö. 2014. Distributed-core view materialization and maintenance for large dynamic graphs. *TKDE* 26(10):2439–2452.

Altaf-Ul-Amine, M.; Nishikata, K.; Korna, T.; Miyasato, T.; Shinbo, Y.; Arifuzzaman, M.; Wada, C.; Maeda, M.; Oshima, T.; Mori, H.; et al. 2003. Prediction of protein functions based on k-cores of protein-protein interaction networks and amino acid sequences. *Genome Informatics* 14:498–499.

Andersen, R., and Chellapilla, K. 2009. Finding dense subgraphs with size bounds. In *WAW*, 25–37.

Batagelj, V., and Zaversnik, M. 2003. An o(m) algorithm for cores decomposition of networks. *CoRR* cs.DS/0310049.

Bhawalkar, K.; Kleinberg, J. M.; Lewi, K.; Roughgarden, T.; and Sharma, A. 2015. Preventing unraveling in social networks: The anchored k-core problem. *SIAM J. Discrete Math.* 29(3):1452–1475.

Carmi, S.; Havlin, S.; Kirkpatrick, S.; Shavitt, Y.; and Shir, E. 2007. A model of internet topology using k-shell decomposition. *PNAS* 104(27):11150–11154.

Cheng, J.; Ke, Y.; Chu, S.; and Özsu, M. T. 2011. Efficient core decomposition in massive networks. In *ICDE*, 51–62.

Chitnis, R. H.; Fomin, F. V.; and Golovach, P. A. 2013. Preventing unraveling in social networks gets harder. In *AAAI*.

Chitnis, R.; Fomin, F. V.; and Golovach, P. A. 2016. Parameterized complexity of the anchored k-core problem for directed graphs. *Inf. Comput.* 247:11–22.

Chwe, M. S.-Y. 2000. Communication and coordination in social networks. *The Review of Economic Studies* 67(1):1–16.

Cui, W.; Xiao, Y.; Wang, H.; and Wang, W. 2014. Local search of communities in large graphs. In *SIGMOD*, 991–1002.

Garcia, D.; Mavrodiev, P.; and Schweitzer, F. 2013. Social resilience in online communities: the autopsy of friendster. In *COSN*, 39–50.

Giatsidis, C.; Malliaros, F. D.; Thilikos, D. M.; and Vazirgiannis, M. 2014. Corecluster: A degeneracy based graph clustering framework. In *AAAI*, 44–50.

Healy, J.; Janssen, J.; Milios, E. E.; and Aiello, W. 2006. Characterization of graphs using degree cores. In *WAW*, 137–148.

Karp, R. M. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, 85–103.

Kitsak, M.; Gallos, L. K.; Havlin, S.; Liljeros, F.; Muchnik, L.; Stanley, H. E.; and Makse, H. A. 2010. Identification of influential spreaders in complex networks. *Nature Physics* 6(11):888–893.

Malliaros, F. D., and Vazirgiannis, M. 2013. To stay or not to stay: modeling engagement dynamics in social graphs. In *CIKM*, 469–478.

Meladianos, P.; Nikolentzos, G.; Rousseau, F.; Stavrakas, Y.; and Vazirgiannis, M. 2015. Degeneracy-based real-time subevent detection in twitter stream. In *ICWSM*, 248–257.

Seidman, S. B. 1983. Network structure and minimum degree. *Social Networks* 5(3):269–287.

Ugander, J.; Backstrom, L.; Marlow, C.; and Kleinberg, J. 2012. Structural diversity in social contagion. *PNAS* 109(16):5962–5966.

Vogiatzis, D. 2013. Influence study on hyper-graphs. In *AAAI*.

Wang, X.; Donaldson, R.; Nell, C.; Gorniak, P.; Ester, M.; and Bu, J. 2016. Recommending groups to users using user-group engagement and time-dependent matrix factorization. In *AAAI*.

Wen, D.; Qin, L.; Zhang, Y.; Lin, X.; and Yu, J. X. 2016. I/O efficient core graph decomposition at web scale. In *ICDE*, 133–144.

Woeginger, G. J. 2001. Exact algorithms for np-hard problems: A survey. In *Combinatorial Optimization*, 185–208.

Wu, S.; Sarma, A. D.; Fabrikant, A.; Lattanzi, S.; and Tomkins, A. 2013. Arrival and departure dynamics in social networks. In *WSDM*, 233–242.

Zhang, Y., and Parthasarathy, S. 2012. Extracting analyzing and visualizing triangle k-core motifs within networks. In *ICDE*, 1049–1060.

Zhang, H.; Zhao, H.; Cai, W.; Liu, J.; and Zhou, W. 2010. Using the k-core decomposition to analyze the static structure of large-scale software systems. *The Journal of Supercomputing* 53(2):352–369.

Zhang, Y.; Yu, J. X.; Zhang, Y.; and Qin, L. 2016. A fast order-based approach for core maintenance. *CoRR* abs/1606.00200.

Zhao, F., and Tung, A. K. H. 2012. Large scale cohesive subgraphs discovery for social network visual analysis. *PVLDB* 6(2):85–96.