Stable Community Detection in Signed Social Networks

Renjie Sun, Chen Chen*, Xiaoyang Wang, Ying Zhang, Xun Wang

Abstract—Community detection is one of the most fundamental problems in social network analysis, while most existing research focuses on unsigned graphs. In real applications, social networks involve not only positive relationships but also negative ones. It is important to exploit the signed information to identify more stable communities. In this paper, we propose a novel model, named stable k-core, to measure the stability of a community in signed graphs. The stable k-core model not only emphasizes user engagement, but also eliminates unstable structures. We show that the problem of finding the maximum stable k-core is NP-hard. To scale for large graphs, novel pruning strategies and searching methods are proposed. We conduct extensive experiments on 6 real-world signed networks to verify the efficiency and effectiveness of proposed model and techniques.

Index Terms—Signed graph, k-core, balanced triangle

1 Introduction

In graph analysis, a key problem is to identify important communities or cohesive subgraphs. The k-core model has been widely adopted to measure the cohesiveness of a subgraph, which requires each user to have at least k friends [1]. The connectivity of the subgraph is also considered in [1]. Most existing research about k-core focuses on unsigned graphs, i.e., only considering friendships [2], [3]. However, friend (i.e., positive neighbor) and foe (i.e., negative neighbor) are ubiquitous [4]. In some social network platforms, such as Epinions[1], users can express trust or distrust towards other users. Ignoring the signed information may fail to characterize the stability of communities.

In signed graph analysis, the concept of balanced triangles serves as a fundamental role [5]. A triangle is a balanced (resp. unbalanced) triangle, if it has odd (resp. even) number of positive edges. The balanced triangle concept is based on the nature that "the friend of my friend is my friend, and the enemy of my enemy is my friend", while unbalanced triangles are against this intuition. As we can observe, for a community or team, unbalanced triangles are not preferred or they can bring unstable factors. In addition, according to the balance theory, a cohesive community should involve few unbalanced triangles [5].

Intuitively, for a stable community in signed graphs, (i) each user should have sufficient number of friends, and (ii) the community is free from unstable factors.

- *Renjie Sun, Chen Chen, Xiaoyang Wang and Xun Wang are with Zhejiang Gongshang University, China. E-mail: renjiesun.zjgsu@gmail.com; {chenc,xiaoyangw, wx}@zjgsu.edu.cn*
- *Ying Zhang is with University of Technology Sydney, Australia. Email:Ying.Zhang@uts.edu.au*
- *Chen Chen is the corresponding author*

1. www.epinions.com

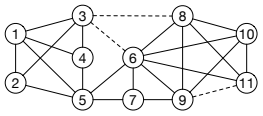


Fig. 1: Motivation example. Dotted lines are negative edges.

In this paper, we propose a new model, named stable k-core, which integrates the properties of k-core model and balanced triangle model. Given a signed graph G, a subgraph S is a stable k-core, if (i) each node in S has at least k positive neighbors, (ii) it does not contain any unbalanced triangles, (iii) it is maximal.

Example 1. *Suppose k=3. Given the signed graph in Figure 1, we can get two stable k-cores, i.e., the subgraphs induced by $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}\}$ and $\{v_1, v_2, v_3, v_4, v_5, v_6, v_8, v_{10}, v_{11}\}$.*

In the literature, different cohesive subgraph models have been proposed for signed graph analysis, such balanced subgraph [6], [7], polarized community [8], [9], k-core based model [10], clique based model [11], etc. However, these models either cannot parameterize the engagement of users in communities (i.e., k), or contain unstable structures (i.e., unbalanced triangles). Due to the properties of unbalanced triangle, they may harm the unity of the community or team. Therefore, the proposed stable k-core model could provide higher quality communities or teams for many applications, such as team formation, community detection, community based promotion, etc. As shown in Example 1, there may be multiple stable k-cores for a given signed graph. In this paper, we focus on finding the maximum stable k-core, which usually preserve the dominant properties of the graph.

Challenges and contributions. The main challenges of this problem lie in the following two aspects. Firstly, we prove the problem of finding the maximum

stable k-core is NP-hard. Secondly, the size of social networks is usually large. To scale for large networks, it is critical to develop efficient searching methods and pruning strategies to accelerate the processing. To the best of our knowledge, this is the first work to investigate the maximum stable k-core problem. Our principle contributions are summarized as follows.

- We formally define the stable k-core model and prove the hardness of the problem.
- Novel pruning strategies and searching methods are proposed in order to scale for large networks.
- We conduct experiments on 6 real signed networks to evaluate the model and proposed techniques.

Related work. In social networks, a community is often represented by a cohesive subgraph. Many cohesive subgraph models are proposed in the literature, e.g., k-core [12], k-truss [13], clique [14], etc. The k-core model is first introduced by Seidman in [1], which has wide spectrum of applications such as influence analysis, community detection, etc [12], [15]. Compared with the research conducted over unsigned graphs, fewer studies pay attention to the problems on signed graphs. Leskovec et al. [16] investigate the classical structural balance theory over online signed networks. Cadena et al. [17] extend the concept of subgraph density for event detection on signed graphs. [18] investigates the influence diffusion procedure over signed networks. In term of communities or cohesive subgraph detection in signed graphs, Giatsidis et al. [10] propose a s-core model by extending the k-core model. In [19], Tsourakakis et al. consider the problem of dense graph detection with negative weights. Note that, a signed graph can be considered as a special case with edge weight $+1$ and -1. Figueiredo et al. [6] investigate the problem of large balanced subgraph detection, and Ordozgoiti et al. [7] propose a more effective and scalable approach for the problem. In [8], Bonchi et al. try to develop efficient algorithms for polarized communities search, and Xiao et al. [9] further study the problem from a local-based perspective. For clique model, Li et al. [11] propose a signed clique model to find densely-connected subgraphs, which have more positive edges but less negative edges. Chen et al. [20] propose a new clique model based on balance properties. A comprehensive survey of signed network mining can be found in [4]. As observed, only s-core leverages the k-core model for signed graph analysis, while it may involve many unbalanced triangles inside, which is also verified by our experiments.

2 Preliminaries

We model a signed graph $G = (V, E)$ as an undirected graph, where V and E denote the set of nodes and edges, respectively. $n = |V|$ and $m = |E|$ are the number of nodes and edges. For $e \in E$, a label $\mathcal{L}(e)$ is associated to e. $\mathcal{L}(e) \rightarrow \{+, -\}$. $\mathcal{L}(e) =$ "$+$" means e is a positive edge (e.g., friendship) and $\mathcal{L}(e) =$ "$-$" means e is a negative edge (e.g., foe relationship). A subgraph $S = (V_S, E_S)$ is an induced subgraph of G, if $V_S \subseteq V$ and $E_S = \{V_S \times V_S\} \cap E$. Given a subgraph S, let $N_S(u) = \{v|(u, v) \in E_S\}$ be the set of neighbors of u in S, and $N_S^+(u) = \{v|(u, v) \in E_S \wedge \mathcal{L}(u, v) =$ "$+$"$\}$ be the set of its positive neighbors. We use $d_S(u) = |N_S(u)|$ and $d_S^+(u) = |N_S^+(u)|$ to denote the degree and positive degree of u in subgraph S, respectively.

Definition 1 (k-core)**.** *Given an unsigned graph G, a subgraph S is the k-core of G, denoted as $C_k(G)$, if (i) $d_S(u) \geq k$ for every $u \in V_S$; and (ii) S is maximal, i.e., any supergraph of S cannot be a k-core.*

A triangle, denoted as $\triangle$, is a cycle of length 3 in the graph. The definitions of balanced and unbalanced triangles are shown as follows.

Definition 2 (Balanced/Unbalanced triangle)**.** *Given a triangle $\triangle$ in a signed graph, $\triangle$ is a balanced triangle, denoted as $\triangle^+$, if it has an odd number of positive edges. Otherwise, $\triangle$ is an unbalanced triangle, denoted as $\triangle^-$.*

Remark 1. *Besides using balanced/unbalanced triangle to classify the stable/unstable structure, users can apply other rules to classify the triangles. As discussed in Section 3.5, the techniques developed in this paper can be easily extended for other cases.*

Definition 3 (stable k-core)**.** *Given a signed graph G and a positive integer k, an induced subgraph S is a stable k-core of G, if it satisfies the following constraints.*

- *Degree: for each node $v \in V_S$, $d_S^+(v) \geq k$;*
- *Stability: S dose not contain any unbalanced triangle;*
- *Maximal: any supergraph of S cannot be a stable k-core.*

As shown in Example 1, there may exist multiple stable k-cores in the signed graph. In this paper, we focus on computing the stable k-core with the largest size, which usually preserves the dominant properties of a network.

Problem Statement. Given a signed graph G and a positive integer k, let $\mathcal{R}$ be the set of all the stable k-cores in G. In this paper, we aim to develop efficient algorithms to identify the **maximum stable k-core** S^*, i.e., $|V_{S^*}| \geq |V_S|$ for any $S \in \mathcal{R}$.

Theorem 1. *Given a signed graph G, the problem of computing the maximum stable k-core is NP-hard.*

Proof: The detailed proof is in Appendix A. □

3 Solution

Due to the hardness of the problem, in this section, we first propose a greedy framework. Then, different optimized techniques are introduced.

3.1 Greedy Framework

To find the maximum stable k-core, we can enumerate different node deletion orders and return the one that

Algorithm 1: Greedy Framework

```
Input  : G: signed graph, k: degree constraint
1 G ← PREPROCESS G ;
2 while △_G^- ≠ ∅ do
3     for each u ∈ Cand(G) do
4         if deleting u can break at least 1 unbalanced
          triangle in △_G^- then
5             compute the score of u ;
6     select the best node u* for deletion ;
7     delete u* and update G ;
8 return G
```

can lead to the largest stable k-core. However, the enumerating space is large. Due to the NP-hardness of the problem, in this paper, we turn to the greedy heuristic. That is, we iteratively remove the best node in the current iteration and update the graph based on the degree constraint, until the remained subgraph does not contain unbalanced triangles. The general greedy framework is shown in Algorithm 1.

$\triangle_G^-$ is the set of unbalanced triangles in current graph G, and $Cand(G)$ means the candidate nodes for deletion. Algorithm 1 just presents a general searching framework, where users can add specific pruning strategies and score functions in Lines 1, 3 and 5 to reduce the searching space, generate candidate nodes and guide the accessing order of the nodes. In the next sections, we will present the detailed methods to finalize the algorithm. In the greedy framework, we iteratively choose the best node for deletion and update G. This is because after deleting a node, other nodes may drop from the community due to the degree constraint. The update procedure can be finished in linear time [12]. In addition, we only compute the score for nodes that can lead to the break down of at least 1 unbalanced triangle.

3.2 Score Function and Candidate Reduction

Given a signed graph G, let $G^+ = (V, E^+)$ be the subgraph that only contains the positive edges, and E^+ is the corresponding edge set. Let $C_k(G^+)$, named **positive k-core**, be the k-core computed over G^+. Then, we have the following lemma holds.

Lemma 1. *The nodes of any stable k-core must be contained in $C_k(G^+)$.*

Proof: $C_k(G^+)$ is the subgraph satisfying the first constraint of stable k-core and it is maximal. Consequently, the nodes of any stable k-core must be included in $C_k(G^+)$. Then, we have the lemma held. □

According to Lemma 1, we can first reduce the searching space by finding the positive k-core. Moreover, the problem of finding the maximum stable k-core is equal to find a set T of nodes from $C_k(G^+)$, such that $C_k(G^+ \setminus T)$ has the largest size and the subgraph induced in the original graph G does not contain any unbalanced triangles. When deleting a node u, it may reduce its positive neighbors' degree and cause a set of nodes to drop from $C_k(G^+)$ in cascade. We call these nodes as the **followers** of u. Since we aim to find the largest community, intuitively, the node with fewer followers is more preferred. We define the score function, named deletion-cost, as follows.

Definition 4 (Deletion-cost)**.** *Let $C_k(G^+)$ be the current positive k-core. The deletion-cost of $u \in V(C_k(G^+))$, denoted as $dc(u)$, equals the number of nodes removed from $C_k(G^+)$ because of the deletion of u, i.e.,*

$$dc(u) = |V(C_k(G^+))| - |V(C_k(G^+ \setminus \{u\}))|$$

To compute the followers of a node, we can traverse through its positive neighbors and remove the nodes that violate the degree constraint. Based on the definition, the best node is the one with the smallest deletion-cost. We randomly assign an order when there is a tie. The minimum deletion-cost equals 1, when the node has no follower. Let $\triangle_k^-$ be the set of unbalanced triangles in the subgraph induced by $V(C_k(G^+))$ in G. Lemma 2 further reduces the candidate space to the nodes in $V(\triangle_k^-)$.

Lemma 2. *There must exist a node set $T \subseteq V(\triangle_k^-)$, the deletion of which can break all $\triangle_k^-$, and the subgraph induced by $V(C_k(G^+ \setminus T))$ is the maximum stable k-core.*

Proof: Given a node set $T_1 \nsubseteq V(\triangle_k^-)$, suppose the deletion of T_1 will lead to the maximum stable k-core. To break an unbalanced triangle, we need to remove at least one of its nodes. Suppose the deletion of T_1 can lead to the deletion of $T_2 \subseteq V(\triangle_k^-)$ and break all the unbalanced triangles. Obviously, the stable k-core obtained by deleting T_1 is no larger than that of directly deleting T_2. It means deleting T_2 will lead to the maximum stable k-core. The lemma is correct. □

3.3 Core-Group based Pruning

In Section 3.2, we reduce the candidate space by only considering the nodes in $V(\triangle_k^-)$. To further reduce the cost, we present the core-group structure.

Definition 5 (Core-group)**.** *Given the positive k-core $C_k(G^+)$ of G, a core-group g of $C_k(G^+)$ is a connected component, in which the degree of each node $v \in V_g$ is k.*

Lemma 3. *Given a core-group g, the deletion of any node in g will cause all nodes in V_g to be deleted. In addition, for $v \in V_g$, $|V_g|$ is a lower bound of $dc(v)$.*

Proof: Since the degree of each node in g is k, the deletion of any node will cause its neighbors' degree reduced by at least one. These neighbors will drop from $C_k(G^+)$ because of the violation of degree constraint. Since the nodes in g are connected, all the other nodes' degree will also be reduced by at least one in cascade. Finally, V_g will be removed from

Algorithm 2: Core-Group Construction

```
Input  : k: degree constraint, C_k(G+): positive k-core
Output : core-groups
1  i = 0 ;
2  for each v ∈ V(△_k^-) do
3      if v is unvisited ∧ d_{C_k(G+)}(v) = k then
4          Q ← {v} ;
5          while Q ≠ ∅ do
6              u ← Q.pop() ;
7              g_i ← g_i ∪ {u} ;
8              for each w ∈ N_{C_k(G+)}(u) do
9                  if w is unvisited ∧ d_{C_k(G+)}(w) = k
                   then
10                     mark w as visited ;
11                     Q ← Q ∪ {w} ;
12         i++ ;
13 return {g_0, g_1, ..., g_{i-1}}
```

$C_k(G^+)$. Obviously, $|V_g|$ is a lower bound of $dc(v)$. Thus, the Lemma holds. □

Algorithm 2 presents the details of building core-groups. In this paper, we only construct the groups containing the candidate nodes (Line 2). We start from the *unvisited* node with degree of k (Line 3) and follow their neighbors with degree of k (Line 9), where g_i denotes the ith core-group constructed. For each traversal, we add the visited nodes into the same group g_i until they have no unvisited neighbors with degree of k (Lines 5-7). Then we start the traversal from the next candidate and build a new core-group. Based on Lemma 3, for each group, we only need to choose one node to compute its deletion-cost. In Lemma 4, we further reduce the cost for the nodes belonging to different connected components.

Lemma 4. *Given the positive k-core $C_k(G^+)$, suppose $C_k(G^+)$ can be divided into a set of connected components $\{c_1, c_2, ..., c_i, ...\}$. If we delete a node $v \in V_{c_i}$, it will not affect the deletion-cost for nodes in V_{c_j}, where $i \neq j$. Moreover, $|V_{c_i}|$ is an upper bound of $dc(v)$ for $v \in V_{c_i}$.*

Proof: Since c_i and c_j are disconnected, deleting any node from c_i will not change the positive degree of nodes in c_j. Moreover, deleting any node from c_i will at most cause the whole component removed from $C_k(G^+)$. Therefore, the lemma is correct. □

Based on Lemma 4, we do not need to recompute the deletion-cost for nodes in c_j, if we have not deleted any node from c_j in the previous iterations. In addition, given the upper and lower bounds obtained in the above lemmas, we can further skip some unpromising nodes.

3.4 Free-Core based Pruning

According to Lemma 1, any stable k-core must be contained in the positive k-core. In a social network, there may exist some densely connected groups, which are free from the unstable structure. Here, we introduce the concept of free-core.

Definition 6 (Free-core)**.** *Given a signed graph G and a positive integer k, let G_f be the subgraph induced by nodes in $V(C_k(G^+)) \setminus V(\triangle_k^-)$. The free-core of G is the positive k-core of G_f.*

Based on the definition, the free-core of G does not contain any unbalanced triangle, and all the nodes have the positive degree no less than k, which means it satisfies the first two constraints in stable k-core.

Lemma 5. *Given a signed graph G, the deletion of nodes in $V(\triangle_k^-)$ will not change the free-core of G, and the free-core must be contained in the maximum stable k-core.*

Proof: Since the free-core is constructed after removing all the nodes in $V(\triangle_k^-)$. Therefore, the deletion of any subset of $V(\triangle_k^-)$ will not change the free-core, i.e., it always satisfies the degree constraint. According to Lemma 2, there must exist a node set $T \subseteq V(\triangle_k^-)$, the deletion of which will lead to the maximum stable k-core. Thus, the free-core must be contained in the maximum stable k-core. □

When we compute the follower of a node u, we need to traverse from u and verify if the visited node can survive from the degree constraint. According to Lemma 5, if certain neighbors of the visited node belong to the free-core, we can stop further checking these neighbors. Moreover, the deletion-cost of a node u equals 1, if it satisfies the constraint in Lemma 6.

Lemma 6. *For a candidate node $u \in V(\triangle_k^-)$, $dc(u) = 1$ if all its positive neighbors are contained in free-core.*

Proof: By assumption, we aim to delete u. We need to check its positive neighbors, since the positive degree of its negative neighbors is not directly affected by the deletion of u. In addition, its positive neighbors all belong to the free-core. According to Lemma 5, the free-core will not change. Therefore, it will only cause the deletion of u itself, i.e., $dc(u) = 1$. □

In each iteration, if there are candidate nodes that satisfy Lemma 6, we can directly delete them, because the deletion-cost of any node is at least 1.

3.5 KST Algorithm

By integrating the score function and all the pruning strategies developed with the greedy framework, we come up with the KST algorithm. In KST algorithm, we reduce the searching space and candidate space by using the positive k-core (Lemma 1) and the unbalanced triangles involved (Lemma 2). We further reduce the computation cost for some candidate nodes if i) they exist in the same core-group (Lemma 3), ii) there exists multiple connected components in the positive k-core (Lemma 4), iii) the candidate node satisfies the free-core constraints (Lemmas 5 and 6).

Time complexity analysis. In the algorithm, we need $O(m)$ time to compute the positive k-core. We use BFS to construct the core-groups and identify the connected components in $O(m+n)$ time. The detection of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3047224, IEEE Transactions on Knowledge and Data Engineering

all unbalanced triangles can be done in $O(m^{1.5})$ time based on the in-memory method [21]. To obtain the free-core, we first remove all the edges in unbalanced triangles and construct the k-core in $O(m)$ time. Then, we iteratively remove the best node in the candidate set $Cand(G)$, until there is no unbalanced triangles. In each iteration, we need to scan all the candidates and it takes $O(m)$ time to update the graph. In the worst case, there will be $|Cand(G)|$ iterations, which takes $O(m|Cand(G)|^2)$ time.

Discussion. In this paper, we divide the triangles into two classes, i.e., balanced and unbalanced triangles. In signed graphs, there are other possible methods to classify the triangles. While, the pruning strategies and searching methods proposed in this paper do not depend on the definitions of balanced and unbalanced triangles. Therefore, the proposed methods can be easily applied for the other cases.

4 Experiments

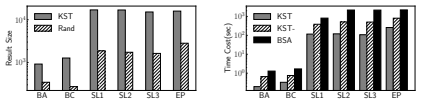
Algorithms. To the best of our knowledge, there is no existing work for the problem studied. Thus, we implement and evaluate the following algorithms.

- **Rand.** In each iteration, it randomly selects a node from the candidate set $V(\triangle_k^-)$ for deletion and updates the graph based on degree constraint, until there is no unbalanced triangle.
- **EXACT.** Use Lemmas 1, 2 and 5 to filter the space, and then leverage the branch and bound framework to find the exact maximum result.
- **BSA.** Use the greedy framework, with deletion-cost as score function and positive k-core as candidates.
- **KST-.** The approach by removing the core-group and free-core based pruning rules from KST.
- **KST.** The approach developed in Section 3.5.

Datasets and workloads. We employ 6 real-world signed networks, i.e., Bitcoinalpha (BA, 7.6k, 14.1K), Bitcoinotc (BC, 6K, 21.5K), Slashdot081106 (SL1, 82.1K 468.5K), Slashdot090216 (SL2, 81.8K, 497.6K), Slashdot090221 (SL3, 77.3K, 500.4K) and Epinions (EP, 131.8K, 711.2K), which are public available. The number after each dataset denotes the corresponding number of nodes and edges. More details of the datasets can be referred to SNAP[2]. The three Slashdot datasets are captured from Slashdot platform in different time periods. k varies from 1 to 10 with 5 as the default value. All the programs are implemented in C++. The experiments are performed on a PC with a Xeon 2.20 GHz CPU and 128 GB RAM.

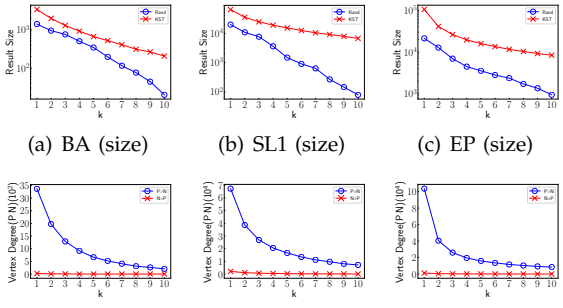
Effectiveness evaluation. To evaluate the effectiveness of proposed techniques, we first report the community size of returned results compared with Rand. Figure 2(a) shows the results on all datasets with the default setting, and Figures 3(a)-3(c) show the results on 3 datasets BA (small-size graph), SL1 (medium-size graph) and EP (large-size graph) by varying k. As

2. https://snap.stanford.edu/data/



(a) Community size (b) Response time

Fig. 2: Experiment results on all the datasets



(a) BA (size) (b) SL1 (size) (c) EP (size)

(d) BA (degree) (e) SL1 (degree) (f) EP (degree)

(g) BA (time) (h) SL1 (time) (i) EP (time)

Fig. 3: Experiment results on BA, SL1 and EP by varying k. (a)-(c) show the result size. (d)-(f) show the degree distribution. (g)-(i) show the response time.

observed, KST returns much larger communities than Rand due to the powerful node selection strategy. As k increases, the result size decreases, since larger k means tighter constraint and leads to smaller result.

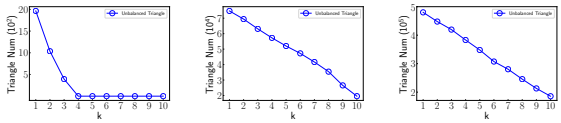
Rationally, for most users in a community, they should have more friends than enemies. Thus, we report the number of nodes with positive degree larger (smaller) than its negative degree in the returned community. The results are shown in Figures 3(d)-3(f) by varying k. As observed, for different k, the number of nodes with negative degree larger than positive degree is very small. It means, even though our model does not set direct constraints for negative degree, it can return a very rational community.

To further evaluate the quality of returned communities, we compare stable k-core with k-core on SL1. For the k-core model, we treat each edge equally. We employ 4 widely used metrics, i.e., diameter, average degree, density and clustering coefficient (CC) [22]. The results are shown in Table 1. The results indicate that our model can return a more cohesive community compared with k-core.

Compare with s-core model [10]. The s-core model also extends the k-core model for signed graph analysis, which requires each node in the subgraph have no less than k positive neighbors and l negative neighbors. Due to the different parameters involved, we vary k from 1 to 10 and set $l = k$. Figure 4 reports the number of unbalanced triangle in s-core on BA, SL1 and EP by varying k. As observed, s-core contains many unbalanced triangles. As discussed,

TABLE 1: Comparison for CD solutions on SL1

Metrics	Diameter	Degree	Density	CC
k-core	7.91	21.38	0.025	0.43
stable k-core	6.72	19.82	0.063	0.51



(a) BA (triangle) (b) SL1 (triangle) (c) EP (triangle)

Fig. 4: Experiment results compared with s-core

these unbalanced triangles may bring unstable factors into the community and harm its unity.

Compare with EXACT. As shown in Figure 5, we report the returned community size and response time compared with EXACT method. Due to the huge time cost of EXACT, we only conduct the experiments on BA dataset. For other datasets, the algorithms cannot be finished in 48 hours. As observed, the return community size is very close, which verifies the effectiveness of proposed greedy strategy. In addition, the developed algorithm is much faster than EXACT.

Case Study. As shown in Figure 6, we conduct case studies on BA and BC. The whole graph is the computed positive k-core. The stable k-core is the subgraph induced by the green nodes. The red nodes compose the unstable parts in positive k-core. As observed, there are a lot of unstable structures. Therefore, only considering unsigned information is not enough to characterize stable communities.

Efficiency Evaluation. To evaluate the efficiency of proposed techniques, we compare KST with KST- and BSA. Figure 2 reports the response time on all datasets with default setting. Figures 3(g)-3(i) show the results on BA, SL1 and EP by varying k. As observed, KST and KST- are much faster than BSA. As more pruning techniques involved, the algorithms runs faster, which verifies the effectiveness of proposed techniques. As k increases, the response time decreases for all methods, since the community size decreases correspondingly.

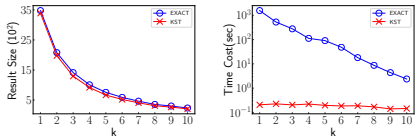
5 Conclusion

In this paper, we investigate the problem of finding stable community in signed graphs. A new model, named stable k-core, is proposed. We prove it is NP-hard to find the maximum stable k-core. To solve the problem, a greedy heuristic framework is developed. Different pruning strategies are further proposed to speedup the computation. Finally, we conduct extensive experiments over 6 real-world signed networks to demonstrate the advantages of proposed techniques.

Acknowledgments. Xiaoyang Wang is supported by NSFC61802345. Chen Chen is supported by ZJNSF LQ20F020007.

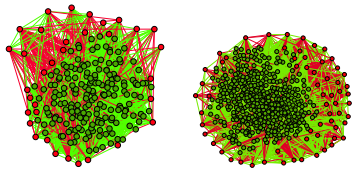
References

[1] S. B. Seidman, "Network structure and minimum degree," *Social networks*, 1983.



(a) BA (size) (b) BA (time)

Fig. 5: Experiment results compared with EXACT



(a) BA (b) BC

Fig. 6: Case study with $k = 10$

[2] Y. Zhang, J. X. Yu, Y. Zhang, and L. Qin, "A fast order-based approach for core maintenance," in *ICDE*, 2017.
[3] F. Morone, G. Ferraro, and H. A. Makse, "The k-core as a predictor of structural collapse in mutualistic ecosystems," *Nature Physics*, 2019.
[4] J. Tang, Y. Chang, C. Aggarwal, and H. Liu, "A survey of signed network mining in social media," *ACM Computing Surveys*, 2016.
[5] D. Cartwright and F. Harary, "Structural balance: a generalization of heider's theory." *Psychological review*, 1956.
[6] R. Figueiredo and Y. Frota, "The maximum balanced subgraph of a signed graph: Applications and solution approaches," *European Journal of Operational Research*, 2014.
[7] B. Ordozgoiti, A. Matakos, and A. Gionis, "Finding large balanced subgraphs in signed networks," in *WWW*, 2020.
[8] F. Bonchi, E. Galimberti, A. Gionis, B. Ordozgoiti, and G. Ruffo, "Discovering polarized communities in signed networks," in *CIKM*, 2019.
[9] H. Xiao, B. Ordozgoiti, and A. Gionis, "Searching for polarization in signed graphs: a local spectral approach," in *WWW*, 2020.
[10] C. Giatsidis, B. Cautis, S. Maniu, D. M. Thilikos, and M. Vazirgiannis, "Quantifying trust dynamics in signed graphs, the s-cores approach," in *SDM*, 2014.
[11] R.-H. Li, Q. Dai, L. Qin, G. Wang, X. Xiao, J. X. Yu, and S. Qiao, "Efficient signed clique search in signed networks," in *ICDE*, 2018.
[12] Y. Zhang, J. X. Yu, Y. Zhang, and L. Qin, "A fast order-based approach for core maintenance," in *ICDE*, 2017.
[13] W. Zhu, M. Zhang, C. Chen, X. Wang, F. Zhang, and X. Lin, "Pivotal relationship identification: The k-truss minimization problem," in *IJCAI*, 2019.
[14] J. Cheng, Y. Ke, A. W.-C. Fu, J. X. Yu, and L. Zhu, "Finding maximal cliques in massive networks," *TODS*, 2011.
[15] Z. Zhou, F. Zhang, X. Lin, W. Zhang, and C. Chen, "K-core maximization: An edge addition approach," in *IJCAI*, 2019.
[16] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Signed networks in social media," in *SIGCHI*, 2010.
[17] J. Cadena, A. K. Vullikanti, and C. C. Aggarwal, "On dense subgraphs in signed network streams," in *ICDM*, 2016.
[18] D. Li and J. Liu, "Modeling influence diffusion over signed social networks," *TKDE*, 2019.
[19] C. E. Tsourakakis, T. Chen, N. Kakimura, and J. Pachocki, "Novel dense subgraph discovery primitives: Risk aversion and exclusion queries," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2019.
[20] Z. Chen, L. Yuan, X. Lin, L. Qin, and J. Yang, "Efficient maximal balanced clique enumeration in signed networks," in *WWW*, 2020.
[21] M. Latapy, "Main-memory triangle computations for very large (sparse (power-law)) graphs," *TCS*, 2008.
[22] Y. Fang, X. Huang, L. Qin, Y. Zhang, W. Zhang, R. Cheng, and X. Lin, "A survey of community search over big graphs," *VLDB Journal*, 2019.