Mining Key Scholars via Collapsed Core and Truss

Conference Paper · August 2019

DOI: 10.1109/DASC/PiCom/CBDCom/CyberSciTech.2019.00063

CITATION
1

READS
40

7 authors, including:

Shuo Yu
Dalian University of Technology
56 PUBLICATIONS **615** CITATIONS
SEE PROFILE

Hayat Dino Bedru
Central South University
13 PUBLICATIONS **78** CITATIONS
SEE PROFILE

Teshome Megersa Bekele
Dalian University of Technology
12 PUBLICATIONS **499** CITATIONS
SEE PROFILE

Liangtian Wan
107 PUBLICATIONS **1,793** CITATIONS
SEE PROFILE

Mining Key Scholars via Collapsed Core and Truss

Shuo Yu, Yuanhu Liu, Jing Ren, Hayat Dino Bedru, Teshome Megersa Bekele, Liangtian Wan, Feng Xia
School of Software, Dalian University of Technology, Dalian 116620, China

***Abstract*—Mining key scholars is an important task in various scenes such as experts finding, collaborator recommendation, etc. With the generalization of collaboration, the co-authorship network has extremely enlarged, which makes this task more difficult to be handled. In this work, we propose two algorithms to mine key scholars based on collapsed k-core/k-truss and closeness centrality. The proposed algorithms mine key scholars mainly from two perspectives, i.e., component segmentation calculation and auxiliary decision metric. The proposed algorithms are irrelevant to network connectivity, which performs with high efficiency. We employ closeness centrality to avoid the derivation of greedy strategy. By comparing with other baseline methods, we find that the two proposed algorithms outperform with higher efficiency and better effectiveness.**

***Index Terms*—Collapsed k-core, collapsed k-truss, key scholars, closeness centrality, network component**

I. Introduction

The rapid development of science and the deepening of scientific research lead to collaborative publications. As the main form of scientific research achievements, academic publications reflect the collaboration relationships. In science and technology management, there exist many significant tasks that require the identification of key scholars, including experts finding and the formulation of science and technology policy, as well as the evaluation of scholars. On the other hand, the departure of key users can seriously undermine network participation, that is, leading to the withdrawal of a large number of other users. Therefore, finding key scholars is an important task in academia. With the increasing scale of academic data, it is time-consuming for scholars to find key nodes in academic networks.

Mining key nodes in the network has begun since long time ago. Current related studies can be divided into two categories, i.e., structural metrics and deletion methods. Key nodes can be identified directly by defining some indices without any modification of the original network. This kind of methods proposes persuasive indices to measure the importance of nodes. Some of the most widely used metrics have already applied, including network metrics (e.g., degree centrality, semi-local centrality, closeness centrality, eigenvector centrality, etc.) and scientific evaluation metrics (e.g., h-index, citation counts, publication number, etc.) [1]. For example, the commonly used degree centrality index takes the degree of nodes in the network as the key index to measure nodes. Random walk can be employed as well to calculate the importance of each node [2]. The other is to calculate the impact of a certain node after deleting it. Some classical methods are proposed such as k-core, k-clique, k-plex, k-truss, etc [3]. Studies about cohesion subgraph have been widely applied in social communication, network analysis, visualization, protein complex analysis, etc [4].

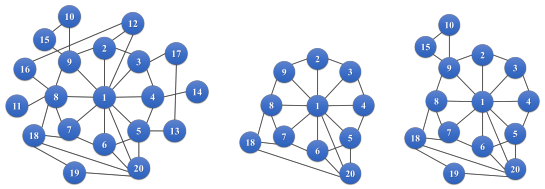
Obviously, methods using structural metrics have the advantages of simplicity and low computational complexity. But its accuracy in mining key scholars is not ideal. Deletion methods mine key nodes by deleting some nodes based on some indicators. The significance of these nodes are judged by the impact on the network after deletion. This kind of methods is relatively complex and the algorithm complexity is generally high. But the accuracy can be significantly improved. Therefore, we integrate the two kinds of solutions to mine key scholars. In our work, we aim at finding a solution for mining key scholars in the academic network with both high effectiveness and accuracy. We propose two algorithms, which are based on collapsed k-core and collapsed k-truss, respectively. To handle the large-scale of academic network, the two proposed methods are proved to be irrelevant to network connectivity and implemented on network components. To better identify the key scholars, an auxiliary decision metric (i.e., closeness centrality) is employed. By comparing with 4 baseline methods for each proposed method, we find that our proposed methods outperform with higher efficiency and effectiveness.

II. Model Design

A. Problem Definition

Consider an undirected co-authorship network $G = (V, E)$, wherein V represents the scholar set and E represents the relationship set. In the network G, a k-core is a subnetwork of G, wherein the degree of each node is no less than k. A k-truss is also a subnetwork of G, wherein each edge with k-truss is involved in more than $k-2$ triangles. Find a subset $V_k \subseteq V$ that satisfies the following two constraint conditions.

1) For a given scholar $s \in V_k$, s owns a high collapsed k-core standard (CCS) or a high collapsed k-truss standard (CTS). Wherein, CCS is defined as the collapsed degree (or the number of followers) in k-core and CTS



(a) Original Network G (b) 3-core (c) 3-truss

Fig. 1. The example of original network, 3-core, and 3-truss.

is defined as the collapsed degree in k-truss (or the number of followers).

2) For a given scholar $s \in V_k$, s owns a high closeness centrality (CC), which is defined as shown in Equation 1.

$$CC_i = \frac{1}{avg_i} = \frac{n-1}{\sum_{i \neq j} d_{ij}} \quad (1)$$

Wherein, avg_i refers to the average degree of i to all of the other nodes and d_{ij} is the distance between i and j.

Conceptually, the definition of k-truss is based on triangle, which is considered as the basic building blocks of the network [5]. For example, in a social network, a triangle means that three friends have a very close relationship, or two people have a common friend. Therefore, k-truss enforces that all edges contain at least $k-2$ triangles, and then enhances that each edge has at least $k-2$ strong relationships. Intuitively speaking, the more common friends two people have in a social network, the closer the relationship between them is. As for k-core, it is defined based on edge connection (i.e., the degree of vertex). Figure 1 shows the difference between k-core and k-truss. It can be proved that calculating CCS or CTS with any k is an NP-hard problem [6].

B. Mining Key Scholars by Collapsed Core

The main idea of mining key scholars is to first find k-cores of the network G. Then calculate the candidate sets of each component of k-cores. By collapsing these k-cores, we can find the best collapsed nodes. If there exist two or more collapsed nodes owning the same followers, the one with higher closeness centrality will be remained. This procedure is proposed to avoid accuracy deviation. According to the definition of closeness centrality, a node with high closeness centrality will lead to collapse easily. Besides, the first procedure of our algorithm, i.e., component partition, makes it possible to calculate the closeness centrality of node. The details of our proposed algorithm based on k-core is shown in Algorithm 1.

C. Mining Key Scholars by Collapsed Truss

Firstly, the k-truss of network G is partitioned into different components. The k-truss calculation has a high probability of dividing the graph into multiple disconnected components, and then each connected component is preprocessed for the first time to calculate the candidate set T. Then we calculate the optimal node of each connected component. Similar like the process of collapsed k-core, when two or more collapsed nodes have the same followers, the one with higher closeness centrality will be remained. The details of our proposed algorithm based on collapsed k-truss is shown in Algorithm 2.

D. Time Complexity Analysis

1) k-core: Suppose there are k components of k-core, the largest component has n nodes and m edges. If the scale of T is t, then the time complexity of the algorithm is $O(s \cdot t \cdot m \cdot \log k)$. All the parameters are far less then the graph scale except s. Therefore, our algorithm will be particularly effective when the calculation is implemented internally.

Algorithm 1: Mining key scholars by collapsed core

Require: G, core model parameter k, the number of key scholars N_s
Ensure: The set of key scholars V_k.
k-core set of G = $kcore$;
$gArray$ = components of $kcore$;
$tArray$ = candidate sets of $gArray$;
P = nodes with degree of k in $kcore$;
ordered set T = nodes with more than 1 neighbor in P;
$V_k = \emptyset$, heap $follower = \emptyset$, $i = 0$;
repeat
  for u in $T \subseteq gArray$ **do**
    delete nodes $u \in T$
    calculate new k-core
    calculate followers of new k-core
    $bestNode$ = node with most followers
    delete the followers of $u \in T$
  end for
  $follower$ = $follower \cup \{bestNode\}$
  delete $bestNode \in kcore$
  $bestSub$ = the component of the top node in $follower$
  delete node in $gArray[bestSub]$
  update $tArray[bestSub]$ and heap $follower$
  V_k = $V_k \cup \{bestNode\}$
until $i < N_s$
return V_k

Algorithm 2: Mining key scholars by collapsed truss

Require: G, truss model parameter k, the number of key scholars N_s
Ensure: The set of key scholars V_k.
k-truss set of G = $ktruss$;
$gArray$ = components of $kcore$;
$tArray$ = candidate sets of $gArray$;
P = edges with support degree = $k-2$ in $ktruss$;
ordered set T = nodes with more than 1 edge in P;
$V_k = \emptyset$, heap $follower = \emptyset$, $i = 0$;
repeat
  for u in $T \subseteq gArray$ **do**
    delete nodes $u \in T$
    calculate new k-truss
    calculate followers of k-truss
    $bestNode$ = node with the most followers
    delete the followers of $u \in T$
  end for
  $follower$ = $follower \cup \{bestNode\}$
  delete $bestNode \in kcore$
  $bestSub$ = the component of the top node in $follower$
  delete node in $gArray[bestSub]$
  update $tArray[bestSub]$ and heap $follower$
  V_k = $V_k \cup \{bestNode\}$
until $i < N_s$
return V_k

2) k-truss: Suppose there are k components of k-truss, the average size of each connected block is $G_{avg} = (n, m)$, and the average size of each connected component candidate set is t. The computational complexity of k-truss is $O(m^{1.5})$. Therefore, the overall computational complexity is $O(s \cdot t \cdot m^{1.5} \cdot \log k)$.

III. Experiments

We implement our algorithms and other 8 baseline methods on two large-scale data sets, i.e., DBLP [1] and APS [2]. DBLP contains publications of Computer Science discipline and APS contains that of Physics. We construct two large networks, respectively. Co-authorship network of APS contains 405,879 nodes and 4,965,916 edges. And DBLP co-authorship network contains 1,384,228 nodes and 6,211,411 edges. We filter out the papers with more than 200 co-authors.

A. Baseline Methods

We compare the proposed collapsed k-core method with the following 4 baseline methods.

- BaselineC [6]: A basic solution for collapsed k-core problem. This method enumerates all the nodes in the k-core set at each iteration. Meanwhile, it deletes every node to find an optimal one. Then update k-core and iterate for s times to find a total of s nodes.
- MaxC: We propose this method as a contrast. Comparing with our proposed method, MaxC uses max node degree instead of closeness centrality as auxiliary decision metric.
- CKC [6]: This is an optimized method of BaselineC, which employs the idea of greedy and pruning.
- OKC: First partition the component, then reduce the candidate sets to only one.

Similarly, we employ the following 4 methods as baseline methods to compare with the proposed collapsed k-truss method.

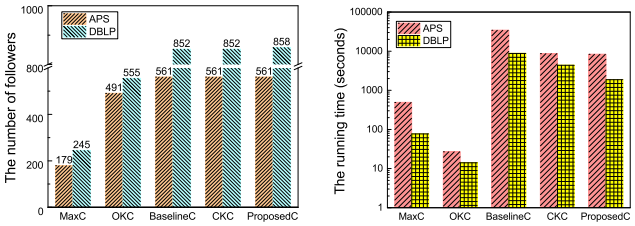
- BaselineT [6]: A basic solution for collapsed k-truss problem. It makes an undifferentiated attempt to find an optimal node in all the nodes in the k-truss at each iteration. Then iterate for s times to find s nodes.
- MaxT: Similar to the MaxC, MaxT selects candidate nodes based on the number of triangles. MaxT selects s nodes from k-truss that constitute the most triangles, and then calculates all the followers.
- CKT [6]: This is an optimized method of BaselineT, which employs the idea of greedy and pruning.
- OKT: First partition the component, then reduce the candidate sets to only one.

B. Experimental Results

According to our methods, there are totally two input parameters, k and N_s. We first set fixed k and fixed N_s to evaluate the computational efficiency and effectiveness. Then we discuss two situations, i.e., ranged N_s with fixed k, ranged k with fixed N_s. We denote our proposed methods as ProposedC (for collapsed k-core) and ProposedT (for collapsed k-truss).

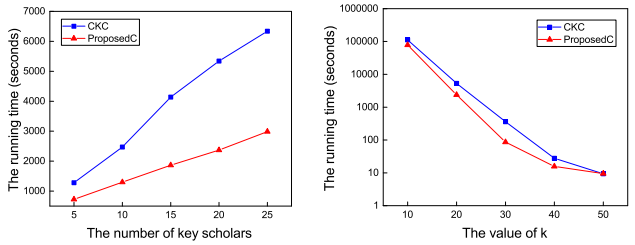
[1]https://dblp.org/

[2]https://www.aps.org/



(a) Followers (b) Runtime

Fig. 2. The experimental results of collapsed k-core.



(a) Fixed k=20 (b) Fixed N_s=20

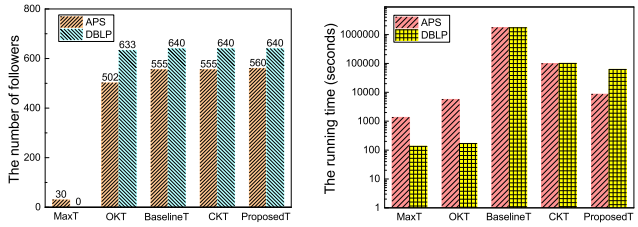
Fig. 3. The running time comparison between CKC and ProposedC with fixed k and fixed N_s, respectively.

TABLE I
THE COMPARISON OF FOLLOWERS NUMBER BETWEEN CKC AND PROPOSEDC.

N_s	Fixed $k = 20$		k	Fixed $N_s = 20$	
	CKC	ProposedC		CKC	ProposedC
5	282	282	10	982	982
10	485	485	20	852	858
15	674	674	30	586	591
20	852	858	40	142	184
25	1,018	1,020	50	185	235

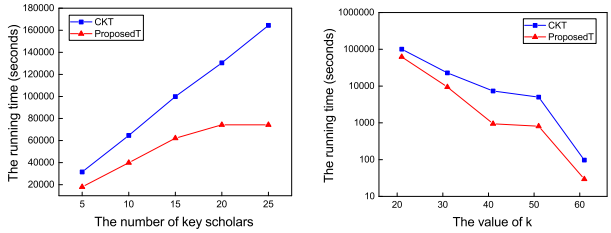
1) Collapsed k-core: We set $k = 20$ and $N_s = 20$ as default values according to previous work [6]. We first compare our proposed k-core method with the other 4 methods (i.e., MaxC, OKC, BaselineC, CKC) to evaluate the mining efficiency and effectiveness. Figure 2(a) shows the number of followers for each method. Followers refer to the nodes collapsed with the deletion of key scholars. The deletion of a more significant scholar will bring more followers. Our algorithm totally mines 561 followers in APS and 858 in DBLP, which outperforms the other baseline methods. Though BaselineC and CKC achieve similar followers, the computational time is much higher than our proposed method (shown in Figure 2(b)). It can be seen that our proposed method consumes the least time. This illustrates that ProposedC is effective and efficient.

Since the mining results is related to k and N_s, we implement several experiments along with them, respectively. We also compare our method with CKC, which performs



(a) Followers (b) Runtime

Fig. 4. The experimental results of collapsed k-truss.



(a) Fixed k=21 (b) Fixed N_s=20

Fig. 5. The running time comparison between CKT and ProposedT with fixed k and fixed N_s, respectively.

best among the baseline methods. We take DBLP data set as case study to verify the effectiveness and efficiency of ProposedC. Experimental results are shown in Figure 3 and Table I. In Figure 3(a), we set $k = 20$ and N_s ranges in $[5, 10, 15, 20, 25]$. It can be seen that our ProposedC achieves much shorter running time when they own similar number of followers. In Figure 3(b), we set $N_s = 20$ and k ranges in $[10, 20, 30, 40, 50]$. The two lines are close in the figure because the y-axis is logarithmic coordinates. Actually ProposedC consumes much less time than CKC does. Table I shows the number of followers. CKC and ProposedC perform with similar results when we fix $k = 20$ and range N_s. But if we fix $N_s = 20$ and range k, it can be seen that ProposedC outperforms with much more followers. This fact indicates that ProposedC outperforms when the network structure becomes more complex. Both of the results show our method performs better.

2) Collapsed k-truss: We set $k = 21$ and $N_s = 20$ as default values corresponding to collapsed k-core. According to the definition of k-truss, the number of triangles formed by each edge is equal or larger than $k - 2$. Therefore, each edge form at least 19 triangles when $k = 21$. The degree of the two connected nodes will be both equal or larger than 20.

Figure 3 shows the overall experimental results of collapsed k-truss. MaxT and OKT consume less time but fails in mining more followers. We also use DBLP data set to verify the effectiveness and the efficiency of CKT and ProposedT. Figure 5 shows the comparison of running time. It is obvious that ProposedT outperforms CKT. Table II shows the comparison of followers. Like the situation in collapsed k-core, ProposedT outperforms especially the structure is more complicated (i.e., k is larger).

TABLE II
THE COMPARISON OF FOLLOWERS NUMBER BETWEEN CKT AND PROPOSEDT.

N_s	Fixed $k = 21$		k	Fixed $N_s = 20$	
	CKC	ProposedC		CKC	ProposedC
5	204	204	21	640	640
10	381	381	31	544	568
15	530	530	41	154	272
20	640	640	51	267	307
25	740	740	61	156	206

C. Discussion

It is more difficult to mine k-truss structure than k-core. This is because that the structure of k-truss is more complex due to its definition. Therefore, we can see that the running time of k-truss based methods are generally longer than k-core based methods, especially in large-scale data sets.

Therefore, we find the optimal solution of multiple components by heap. Each update only needs to be performed under each connected component, which significantly reduces the algorithm complexity. In addition, the proposed algorithms employ an auxiliary metric, i.e., closeness centrality to assist decision-making, which also improves the effectiveness of the algorithms.

IV. CONCLUSION

In this work, we propose two methods to mine key scholars in academia. This method employ the collapsed k-core and k-truss as well as closeness centrality to recognize key nodes in the co-authorship networks. The two proposed methods are proved to be irrelevant to network connectivity. Therefore, we update the iteration results within certain network components to improve computational efficiency. Moreover, we also employ closeness centrality to guide the choice when some nodes have same number of followers. By comparing with other 8 baseline methods, our proposed methods show better performance with both higher efficiency and greater effectiveness.

REFERENCES

[1] X. Kong, Y. Shi, S. Yu, J. Liu, and F. Xia, "Academic social networks: Modeling, analysis, mining and applications," *Journal of Network and Computer Applications*, 2019.

[2] S. Yu, J. Liu, Z. Yang, Z. Chen, H. Jiang, A. Tolba, and F. Xia, "Pave: Personalized academic venue recommendation exploiting co-publication networks," *Journal of Network and Computer Applications*, vol. 104, pp. 38–47, 2018.

[3] S. Yoon, A. Goltsev, and J. Mendes, "Structural stability of interaction networks against negative external fields," *Physical Review E*, vol. 97, no. 4, p. 042311, 2018.

[4] F. Morone, G. Del Ferraro, and H. A. Makse, "The k-core as a predictor of structural collapse in mutualistic ecosystems," *Nature Physics*, vol. 15, no. 1, p. 95, 2019.

[5] A. R. Benson, D. F. Gleich, and J. Leskovec, "Higher-order organization of complex networks," *Science*, vol. 353, no. 6295, pp. 163–166, 2016.

[6] F. Zhang, C. Li, Y. Zhang, L. Qin, and W. Zhang, "Finding critical users in social communities: The collapsed core and truss problems," *IEEE Transactions on Knowledge and Data Engineering*, 2018.