1 Methodology

1.1 Algorithm Pseudo-code

Algorithm 1 BETCS

Input: a graph $G(V, E)$, a query node q, i-hop i, restart probability c, number of levels l, k-truss k.
Output: the k-truss community D containing q.

1. Identify the rough i-hop subgraph containing D based on q in G.
2. Measure the proximity values of nodes to q in the i-hop subgraph with c.
3. Obtain the score vector r, and r_j is the proximity of node j to q.
4. Grade nodes based on r, and lower levels indicate higher proximity.
5. Construct an enhanced graph $G_{en}(V, E \bigcup E_{en})$ based on E_{en} obtained by connecting nodes at a low level with nodes at a level above them.
6. Perform the edge-enhanced k-truss community search based on G_{en} to obtain the desired community D containing q.

As shown in Fig. 1, assuming that the desired community consists of nodes 6, 7, 8, 9, 10, 11, and 12, and that the query node is node 8, we performed a K-truss community search and BETCS on the original graph separately. From the comparison of results, we can see that the community found by K-truss only contains 5 nodes, and nodes 11 and 12 become two isolated nodes due to the lack of truss connections between them and node 8, even though they are well-connected to node 8 and belong to the community in the original graph. Nevertheless, the community obtained by BETCS not only preserves the connectivity of nodes 11 and 12 with node 8, but also enhances their connectivity with other nodes in the community, which makes these nodes a more dense and cohesive community. From this example, we can conclude that our BETCS can address the fragmentation issue from which the K-truss method suffers.

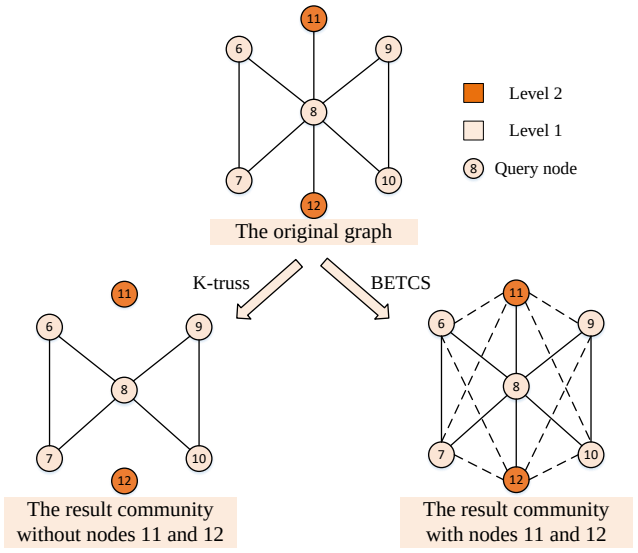


Fig. 1 The community found by K-truss vs the community found by BETCS. The latter contains more nodes that are intuitively close to the query node 8.

1.2 Time Complexity

Next, the time complexity of BETCS is analyzed. Initially, we spend $O(|E|)$ time identifying the rough i-hop subgraph H containing the desired community, where $|E|$ is the edge number of the original graph G. Then, we measure the proximity values of nodes to the query node by using RWR as an example in this paper and solve it using an iterative-until-convergence method. The time required is squared with the number of nodes n in H and linear with the number of iterations and it can achieve convergence in a few iterations. Hence, we spend $O(n^2)$ time measuring the proximity values of nodes in H. Moreover, we usually divide the nodes in H into five levels and then enhance edges between different levels. Therefore, we spend $O(n)$ time finishing this step. Ultimately, the edge-enhanced k-truss community search is performed on H. So we spend $O\left(\sum_{(x,y)\in E_{en}} \min\{d(x), d(y)\}\right)$ time finishing the truss decomposition and $O(d_{max}|UKC|)$ time finishing the query processing, where E_{en} denotes the enhanced edge set in H, $|UKC|$ denotes the edge number in the union of k-truss communities (UKC), and d_{max} denotes the maximum node degree in UKC [1], respectively. To summarize, the total time complexity of BETCS is $O(|E| + n^2 + n + (\sum_{(x,y)\in E_{en}} \min\{d(x), d(y)\} + d_{max}|UKC|))$.

2 Experiment

2.1 Influence of parameters

In this subsection, we test the sensitivity of BETCS with regard to four parameters, namely i of the i-hop subgraph, restart probability c of RWR, number of levels l of grading nodes, and k of k-truss. We are curious about the changes in F-score with different values of i, c, l, k. Hence, we study the connections of the F-score value with these four parameters successively in the DBLP network.

For BETCS, parameter i determines the scope of the desired community containing the query node. When i is small, the scope is narrow, which means that it will cost little time to discover the community, but lead to a low value of F-score. As some nodes that belong to the ground-truth community may not be included in the narrow scope. When i increases, the scope increases, which leads to more time to discover the community. When i is large, it also leads to a low value of

F-score, because the i-hop subgraph will contain numerous irrelevant nodes.

Fig. 2(a) shows the change in F-score when varying i from 1 to 10, which verifies the above conjecture. The value of F-score is the minimum when $i = 1$. When $i = 2$, the F-score value reaches the maximum. When i continues to increase, the F-score value decreases. When $i \geqslant 8$, it becomes almost stable, because no more nodes will be contained in the final identified community when using BETCS on such a large subgraph.

The value of c denotes the restart probability in RWR. From the above introduction to RWR, we know that the random walker returns to the starting node with c at each step. An iterative-until-convergence method is adopted to solve the RWR problem. Therefore, c determines the number of iterations, a larger c indicates fewer iterations and worse effectiveness. Hence, we consider the change in F-score when varying c from 0.1 to 0.5.

As shown in Fig. 2(b), the values of F-score are almost stable when c changes. When $c = 0.15$, the BETCS algorithm has the optimal performance. After that, the F-score value slightly decreases when increasing c.

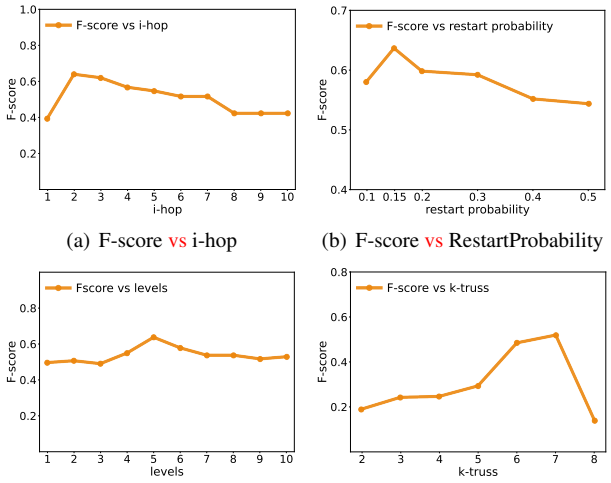
Parameter l determines how many levels the nodes are divided into. A larger l means more levels and more edges are enhanced as we enhance edges by connecting nodes at a lower level to nodes at a level above them.

Fig. 2(c) shows the relationship between F-score and l. It can be seen that the F-score values are almost stable when varying l from 1 to 10. When $l = 5$, the BETCS algorithm achieves optimal performance. It should be noted that no edges are enhanced when $l = 1$.

Initially, from the above introduction to k-truss, it is easy to see that k determines the size and cohesiveness of the final identified community. After edges are enhanced, the i-hop subgraph is more cohesive and dense. Therefore, we may need a larger k to find the desired community in the enhanced i-hop subgraph than in the original graph.

As shown in Fig. 2(d), the values of the F-score value vary greatly when varying k from 2 to 8. When varying k from 2 to 7, the F-score value monotonically increases with k. Therefore, when $k = 7$, the BETCS algorithm achieves optimal performance. When $k = 8$, it should be noted that we set the F-score values of those enhanced subgraphs without 8-truss to 0. Hence, the F-score value is extremely low when $k = 8$.

According to the sensitivity results of the above four parameters of BETCS, it can be seen that a good result can be achieved with i between 2 and 3, c between 0.1 and 0.2, l between 4 and 6, and k between 6 and 7. Hence, we recommend setting $i = 2$, $c = 0.15$, $l = 5$, and $k = 7$ as default parameters when performing BETCS on large networks like DBLP.



(a) F-score vs i-hop (b) F-score vs RestartProbability

(c) F-score vs Levels (d) F-score vs k-truss

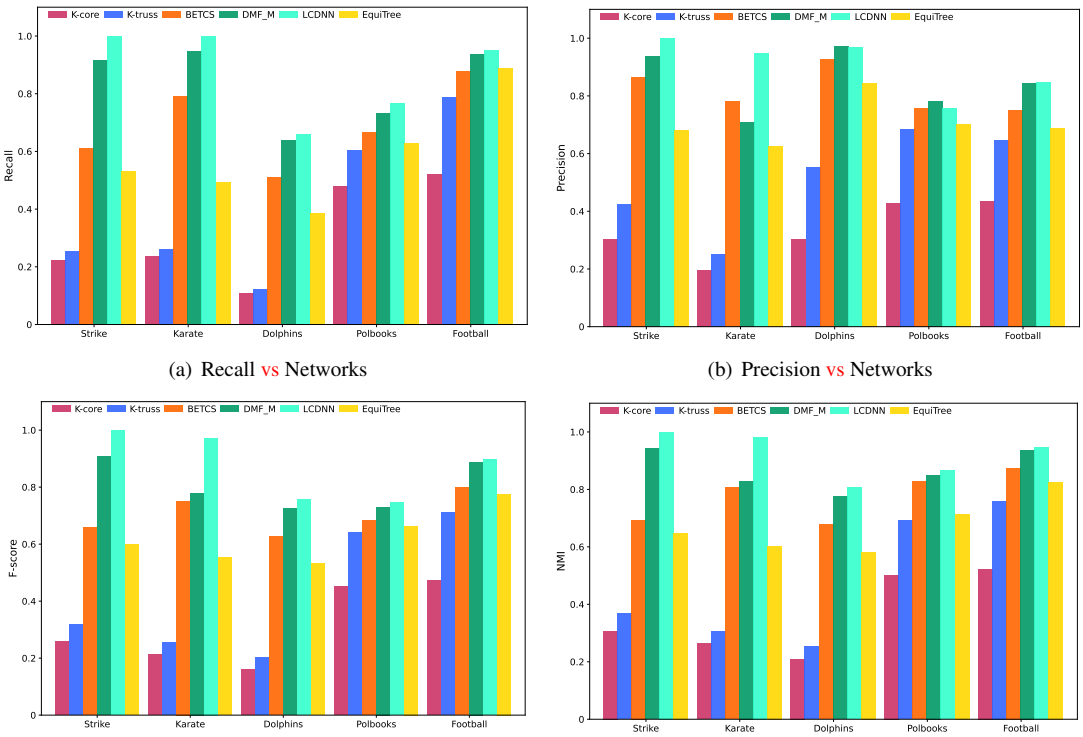
Fig. 2 Sensitivity of four parameters i-hop i, restart probability c, levels l, k-truss k.

2.2 Evaluation on Small-scale Networks

Then, the effectiveness of these six algorithms is evaluated on five small networks with ground-truth communities. For these five small networks, each node of the network is selected as the query node for community search. After adjusting the parameters of each algorithm, we perform these five algorithms on the five small-scale networks and calculate the average value of *recall*, *precision*, $F-score$ and NMI of the corresponding communities. The results are shown in Fig. 3.

It takes little time to perform these six algorithms on the five small networks, which means that taking time as a metric is meaningless in these small networks. Hence, we replace *time* with *recall* and *precision* as newly added metrics. Similar to $F-score$ and NMI, they are also positive metrics that measure the correspondence between the detected community and ground-truth community, and higher values indicate better performance to a certain extent.

In Fig. 3, we can see that the performance of BETCS is the third best and far exceeds that of K-truss and EquiTree, especially in the first three networks, namely Strike, Karate and Dolphins networks. In detail, the F-score value of BETCS is more than 106% and 10% higher than that of K-truss and EquiTree in Strike network, 193% and 35% higher than that of K-truss and EquiTree in Karate network, 208% and 17% higher than that of K-truss and EquiTree in Dolphins network. Besides, in the other two networks, namely Polbooks and Football networks, BETCS improve nearly 7%, 13% and 3%, 3% over K-truss and EquiTree in terms of F-score, respectively. In addition, the reason LCDNN achieves the best performance among these six algorithms in the small-scale networks is that it extends the community from the perspective of each single node, which is more in line with the characteristics of small-scale networks. From the description above, we can see that BETCS is able to address the fragmentation problem faced by methods based on truss structures such as K-truss and EquiTree. However, since fragmentation problem is not as prevalent in small networks as it is in large networks, the BETCS method does not perform as well in



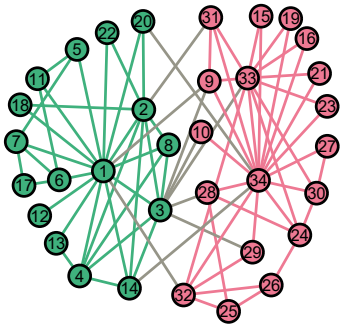
(a) Recall vs Networks

(b) Precision vs Networks

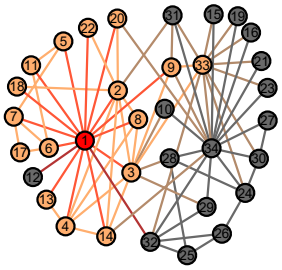
(c) F-score vs Networks

(d) NMI vs Networks

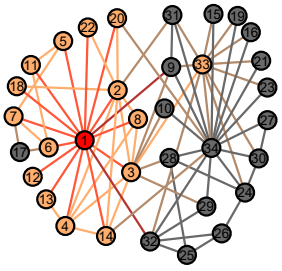
Fig. 3 Accuracy of six algorithms on small-scale networks in terms of Recall, Precision, Average F-score and NMI.



(a) Original Karate network (2 communities, 34 nodes, and 78 edges).



(b) The communtiy identified by k-truss (k = 3) community search with "1" as the query node (F-score is 0.91).



(c) The community identified by BETCS (k = 5) with "1" as the query node (F-score is 0.94).

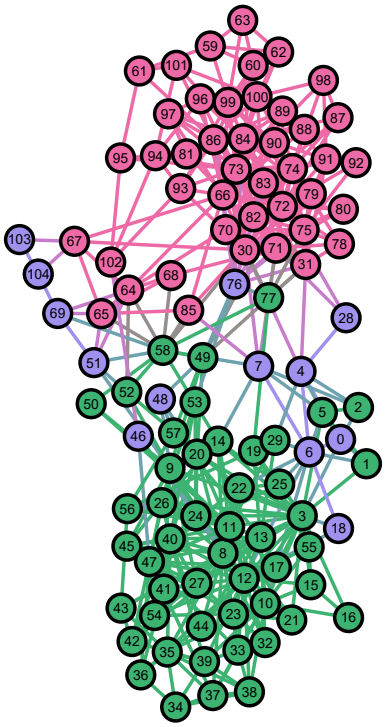
Fig. 4 Performance of k-truss and BETCS algorithms on the Karate network.

small networks as it does in large networks.

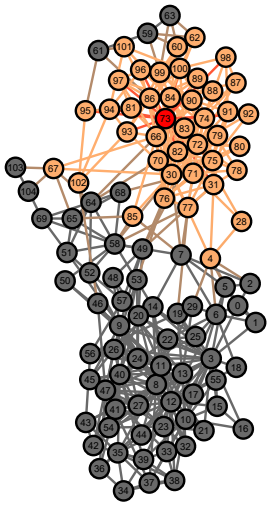
2.3 Case study

To intuitively show the effectiveness of BETCS, we compare BETCS with the k-truss community search on three real-world networks with ground truth communities. Comparing the real community with the community identified by BETCS, we know that the more similar they are, the better the effectiveness of BETCS. Comparing the community identified by the k-truss community search with the community identified by BETCS, we know that the fragmentation issue is well addressed by BETCS to a certain extent. In the original network, we denote different real communities with different colors. For example, in Zachary's karate club network, green represents one community and pink represents the other. In the detected network, we mark the query node in red, the identified communities in orange, and the unidentified communities in gray. We should be aware that the query node in red belongs to the identified communities in orange as well. The first two real-world networks is available from *https://networkdata.ics.uci.edu/index.php*. The Cora network is available from *https://linqs-data.soe.ucsc.edu/public/lbc*. Next, we detail the three case studies.

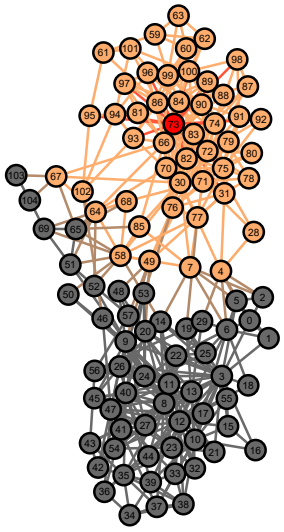
Zachary's karate club network consists of 34 nodes and 78 edges. Each node denotes a member of the club, and each edge denotes a tie between two members of the club [2]. In Fig. 4(a), the original Karate network consists of two com-



(a) Original Polbooks network (3 communities, 105 nodes, and 441 edges).



(b) Communtiy identified by k-truss (k = 3) community search with "73" as the query node (F-score is 0.88).



(c) Community identified by BETCS (k = 3) with "73" as the query node (F-score is 0.92).

Fig. 5 Performance of k-truss and BETCS algorithms on the Polbooks network.

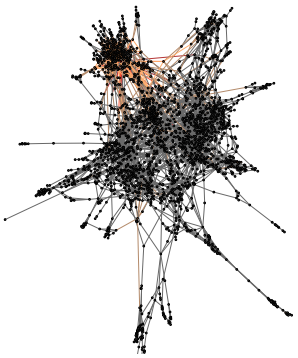
munities. Next, we perform the k-truss community search algorithm and BETCS algorithm based on query node "1" which belongs to the green community with 16 nodes. In Fig. 4(b), the community identified by k-truss (k = 3) community search is denoted by orange nodes with "1" as the query node. It has 1 less real node and 2 more extra nodes than the real community, so the F-score value is 0.91. In Fig. 4(c), after setting i = 2, c = 0.15, and l = 5, the community identified by BETCS (k = 5) is denoted by orange nodes with "1" as the query node. It has 1 less real node and 1 more extra node than the real community, so the F-score value is 0.94.

Books about US politics network consists of 105 nodes and 441 edges. Nodes denote books, and edges denote frequent co-purchasing of books by the same buyers [3]. In Fig. 5(a), the original Polbooks network consists of three communities. Next, we perform the k-truss community search algorithm and BETCS algorithm based on query node "73" which belongs to the pink community with 43 nodes. In Fig. 5(b), the community identified by k-truss (k = 3) community search is denoted by orange nodes with "73" as the query node. It has 6 less real nodes and 4 more extra nodes than the real community, so the F-score value is 0.88. In Fig. 5(c), after setting i = 2, c = 0.15, and l = 5, the community identified by BETCS (k = 3) is denoted by orange nodes with "73" as the query node. It has 1 less real node and 7 more extra nodes than the real community, so the F-score value is 0.92.

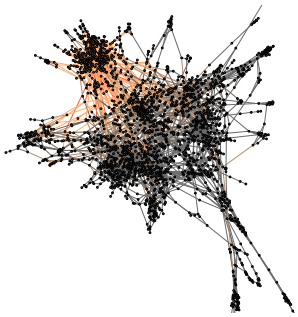
The Cora network consists of 2708 nodes and 5429 edges. Each node denotes one paper, and each edge denotes the connection between a citing paper and a cited paper. In Fig. 6(a), the original Cora network consists of seven communities. Next, we perform the k-truss community search algorithm and BETCS algorithm based on query node "35" which belongs to the blue community with 418 nodes. (For brevity, the node labels are not shown). Comparing the orange community in Fig. 6(b) with that in Fig. 6(c), we observe that the latter contains more nodes and is closer to the real blue community in the original network, thus intuitively having a better performance. The details are as follows. In Fig. 6(b), the community identified by k-truss (k = 3) community search is denoted by orange nodes with "35" as the query node. It has 247 less real nodes and 9 more extra nodes than the real community, so the F-score value is 0.57. In Fig. 6(c), after setting i = 2, c = 0.15, and l = 5, the community identified by BETCS (k = 3) is denoted by orange nodes with "35" as the query node. It has 100 less real nodes and 108 more extra nodes than the real community, so the F-score value is 0.75. Through specific analysis, we can see that BETCS can indeed search the more accurate community than k-truss community search.

According to the three case studies above, the following conclusions can be drawn. (1) BETCS can address the fragmentation issue faced by k-truss community search to a certain extent. (2) BETCS can search more accurate communities than k-truss community search in different networks.

(a) Original Cora network (7 communities, 2708 nodes, and 5429 edges).



(b) Communtiy identified by k-truss ($k = 3$) community search with "35" as the query node (F-score is 0.57).



(c) Community identified by BETCS ($k = 3$) with "35" as the query node (F-score is 0.75).

Fig. 6 Performance of k-truss and BETCS algorithms on the Cora network.

References

1. Xin Huang, Hong Cheng, Lu Qin, Wentao Tian, and Jeffrey Xu Yu. Querying k-truss community in large and dynamic graphs. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1311–1322, 2014.
2. Jian Pei, Yannis Manolopoulos, Shazia Sadiq, and Jianxin Li. Correction to: Database systems for advanced applications. In *International Conference on Database Systems for Advanced Applications*, pages C1–C1. Springer, 2018.
3. ZeJun Sun, Bin Wang, JinFang Sheng, ZhongJing Yu, RongPei Zhou, and JunMing Shao. Community detection based on information dynamics. *Neurocomputing*, 359:341–352, 2019.