Accelerating Community Detection by Using K-core Subgraphs

Chengbin Peng
CEMSE
King Abdullah University of Science and Technology
Thuwal, Saudi Arabia
chengbin.peng@kaust.edu.sa

Tamara G. Kolda
Sandia National Laboratories
Livermore, CA, USA
tgkolda@sandia.gov

Ali Pinar
Sandia National Laboratories
Livermore, CA, USA
apinar@sandia.gov

ABSTRACT
The K-core of a graph is the largest subgraph within which each node has at least K connections. The key observation of this paper is that the K-core may be much smaller than the original graph while retaining its community structure. Building on this observation, we propose a framework that can accelerate community detection algorithms by first focusing on the K-core and then inferring community labels for the remaining nodes. Finally, label propagation and local modularity maximization algorithms are adopted, for their speed and quality, to optimize the community structure of the whole graph. Our experiments demonstrate that the proposed framework can reduce the running time by up to 80% while preserving the quality of the solutions. Theoretical investigations via the likelihood function based on stochastic blockmodels support our framework.

Categories and Subject Descriptors
H.2.8 [Database Applications]: Data mining; E.1 [Data Structures]: Graphs and networks

Keywords
K-core, Community Detection, Modularity

1. INTRODUCTION
The community structure of a graph reflects the connectivity between different nodes. Within each community, nodes are more likely to be connected to each other than to nodes from different communities. Many real world networks have inherent community structures, including social networks, transportation networks, biological networks, etc. The community structure also plays an important role in network dynamics, such as the spreading of diseases. If each node is allowed to exist in only one community, the community structure is called non-overlapping. In this work, we consider the non-overlapping case only.

There are many algorithms developed for such community detection problems. Some algorithms use stochastic blockmodels as prior assumptions, and use Bayesian inference to maximize the likelihood. This approach is similar to spectral algorithms in principle. Recently, algorithms for large-scale community detection have been devised as well, with linear or near-linear time complexity. However, even those linear-time algorithms become slow when detecting communities for large graphs in the big data era because the coefficient omitted in the time complexity is not negligible.

The fundamental observation of this paper is that community structure is preserved in the K-core subgraph even for moderate values of K. The K-core is defined as the largest subgraph in which each node has at least K edges. Algorithms for finding the K-core are efficient and amenable for parallelization. Some papers discuss the existence condition of a K-core subgraph. Some study the relationship between the K-core and the percolation dynamics, and it is worth noting that the bond percolation is quite similar to the label propagation in community detection. Recent work has also found some connections between the K-core structure and the robustness of communities.

In this work, we propose a framework to accelerate community detection algorithms with the detection quality preserved by utilizing the K-core subgraph. The acceleration framework is composed of three steps.

(1) Extract the K-core subgraph, which makes the problem size much smaller.
(2) Apply a community detection algorithm (the main algorithm) on the K-core, where the running time reduces significantly.
(3) Apply a fast algorithm to find the community labels for the whole graph utilizing the results from the second step.

In step two, the main algorithm can be any community detection algorithm, and it ensures the quality for the community configuration in K-core. If the main algorithm is to maximize the likelihood of a stochastic blockmodel, under some assumptions, we prove that whether the community configuration is inferred from the K-core or from the whole graph, it is expected to be invariant for nodes in K-core with a high probability.
In the third step, we devise a simple and fast approach to determine the community labels for peripheral nodes outside the $K$-core effectively, and then optimize the labels of nodes for the whole graph.

In the experiments on real data, we use different $K$’s for comparison, which includes the case of running the main algorithm on the whole graph ($K = 1$). Several main algorithms are used to demonstrate how our framework reduces the running time significantly without sacrificing the detection quality. These algorithms include Bayesian inference [12], label propagation [23], a fast multi-scale approach [10], the Louvain method [1], a modularity maximization method [21], and a general purpose spectral clustering method [32].

The rest of the paper is organized as follows. Section 2 introduces some recent efforts in accelerating community detection problems. Section 3 provides the algorithmic framework of our approach. Section 4 presents some experimental results in terms of the quality and the running time. Section 5 discusses some theoretical supports based on the stochastic blockmodels, and Section 6 concludes the paper.

2. RELATED WORK

Many researches have devoted to community detection algorithms [17, 5]. However, as the problem size becomes larger and larger in real world applications, fast algorithms producing high quality results are in demand. One strategy is to split the graph into very small subgraphs and optimize each small problem locally [28, 13], in which case the quality is usually compromised. Alternatively, another strategy is to include additional phases for global optimization [16, 1, 7], but the running time becomes longer. Multi-threaded algorithms are introduced as well [21, 2, 33]. Some other fast algorithms rely on additional information, such as node attributes [26, 30].

Recently, several metrics for evaluating community structure are also developed such as robustness [18] and influence [8], but modularity [3, 22] remains one of the most popular metrics.

In this paper, we propose a framework to make many community detection algorithms faster with similar or better modularity for the output.

3. ALGORITHM

Intuitively, our framework reduces the problem size for the main community detection algorithm, and then uses the output as the initial value for a fast algorithm to solve the whole problem. We expect to have time reduction and quality preservation compared to running the whole problem solely by the main algorithm.

Specifically, our framework comprises three steps. The first step is to reduce the whole graph to a $K$-core. The second step uses a main algorithm to find the community structure for nodes in the $K$-core. The third step is to use the community structure of the $K$-core as the initial value and find the community structure of the whole graph by a fast algorithm. Algorithm 1 is the pseudocode of the framework, where $G$ is the original graph, $g$ is a vector of community labels, $K$ is some integer, $G_K$ is the $K$-core, and $g_K$ contains community labels for the nodes in $K$-core.

**Algorithm 1** Accelerating Community Detection via $K$-core Subgraphs

1: input Graph $G$, Parameter $K$
2: output Labels $g$
3: (1) $G_K \leftarrow Kcore(G, K)$
4: (2) $g_K \leftarrow CommunityDetection(G_K)$
5: (3) $g \leftarrow Recover(G, g_K)$
6: return $g$

3.1 Find the $K$-core Subgraph

There are many algorithms available to find the $K$-core of a graph [28, 13]. A standard approach [14] is to traverse all the nodes and remove those connected by less than $K$ edges. The edges connected to the removed nodes are removed as well. The traversal and removal may be repeated multiple times, until all the remaining nodes have degrees at least $K$. The pseudocode is presented in Algorithm 2.

**Algorithm 2** Algorithm to Find the $K$-core Subgraph

1: input Graph $G$, Parameter $K$
2: output Subgraph $G_K$
3: while $G_K$ is not a $K$-core do
4: Remove nodes with degree less than $K$ (including their incident edges)
5: Update the node degrees for the remaining nodes
6: end while
7: return $G_K$

3.2 Apply a Community Detection Algorithm on the $K$-core

Applying the community detection algorithm to the $K$-core is the main part in our framework, and any community detection algorithm can be used here. The biggest performance gains will be for methods with high quality and a high time complexity. The main algorithm detects the community structure only for nodes in the $K$-core, which is the main reason for the running time reduction.

Here we discuss briefly the effect of our framework on different algorithms. For algorithms with high quality and fairly fast running time, our framework can reduce the running time significantly with the quality preserved. Those algorithms include the Louvain method [1], Martelot’s method [16], and modularity optimization [21]. For very fast algorithms such as label propagation [23], our framework does not improve the running time very much but can improve the quality. Some algorithms like the Bayesian method [12] and spectral clustering [32] need to specify a community number before running, but our framework can increase the community number from the predefined value and to improve the quality.

3.3 Recover the Community Structure for the Whole Graph

When the community structure of the $K$-core ($g_K$) is known, the third step is to recover the node labels for all the nodes including the peripheral ones. With the knowledge from the
As widely used for evaluating the quality of communities, modularity is defined as
\[
Q = \sum_r (e_r - a_r^2),
\]
where \( r \) is an index for communities, \( m \) is the number of edges, \( A \) is the adjacency matrix of \( G \), \( g_j \) is the community label of node \( j \), and \( e_r = \frac{1}{2m} \sum_{ij} A_{ij}\delta(g_i, r)\delta(g_j, r) \), \( a_r = \frac{1}{2m} \sum_i A_{ij}\delta(g_j, r) \). \( \delta \) is the Kronecker function.

Therefore, when a node \( i \) is moved from community \( r \) to community \( s \) \((r \neq s)\), the change of modularity is as follows.
\[
\Delta Q = Q^{(2)} - Q^{(1)} = \sum_i [(e_r^{(2)} - a_r^{(2)})^2 - (e_r^{(1)} - a_r^{(1)})^2] = [(e_r^{(1)} - d_r) - (a_r^{(1)} - d_s)^2 + (e_s^{(1)} + d_s) - (a_s^{(1)} + d_s)^2] = (-d_r + d_s) + 2d(a_r^{(1)} - a_s^{(1)}) - 2d^2. \tag{2}
\]
where \( d_r = \frac{1}{2m} \sum_j (A_{ij} + A_{ji})\delta(g_j, r) + A_{ii} \) and \( d = \frac{1}{2m} \sum_j A_{ij} \). For undirected graphs without self-loops, \( d_r = \frac{1}{m} \sum_j A_{ij} \).

In local modularity optimization, we traverse all the nodes for multiple rounds, and change the community label for each node correspondingly to have a maximum possible \( \Delta Q \) if it is larger than zero.

### 3.4 Time Complexity

The running time of the algorithm in the third step grows linearly with the size of the graph.

Let \( b \) be some constant. Let the graph size be \( N \), and a corresponding \( K \)-core subgraph size be \( N_K \). If a community detection algorithm has time complexity \( O(N^b) \) to find the community structure, by our framework, the complexity can be reduced to \( O(N_K^b + N) \). Therefore, the speedup is \( \left(\frac{N_K}{N}\right)^b \) asymptotically.

### 4. EXPERIMENTS

In this section, we demonstrate that, on real world data, our approach can reduce the computing time significantly, while the quality of the detected communities changes little and sometimes even becomes better. We use modularity as the quality measurement \[3\][22].

#### 4.1 Data and Algorithms

We use the real world data from the Stanford Large Network Dataset Collection at SNAP\[3\], as shown in Table 1. For the directed graph wiki-Vote, we treat the edges as undirected.

To demonstrate the effectiveness of our framework, we incorporate several community detection algorithms, and compare the resulting modularity and the running time under different choice of \( K \). When \( K = 1 \), the result is equivalent to running the main algorithm on the whole graph without our framework.

- **Louvain method** \[1\]: initializes each node as a single community, and shift the community label of each node according to the modularity gain, until the labels converge. Then, it considers each community as a node to run the main algorithm on it.

\[\text{Stanford Network Analysis Platform, http://snap.stanford.edu/ncp/}\]
merge some of them again according to the modularity gain.

- **Martelot’s method** [16]: a multi-scale algorithm with two phases, optimizing a global and a local criterion respectively. In our experiment, we use the modularity criterion.
- **Modularity optimization** [21]: an agglomerative method that merges nodes into bigger and bigger communities hierarchically, using the modularity criterion.
- **Bayesian method** [12]: uses a variational approach to solve the parameter inference problem.
- **Spectral clustering** [32]: considers the graph as a similarity matrix, and solves a data clustering problem where each cluster is a community. It contains two phases. First, it maps the data onto a lower dimensional space formed by eigenvectors of the Laplacian matrix, and second, it uses k-means to cluster the dimension reduced data. It runs relatively fast with acceptable qualities on some data sets, but the user needs to pick an appropriate number of communities in advance.
- **Label propagation** [23]: initializes each node as a single community and shifts the community label of each node to the most frequent neighboring community label, until the labels converge. It runs quite fast, but the quality is usually compromised. Also, the order for nodes to update is important, so in the experiments we repeatedly run for five times using random orders, and the results are averaged.

In the experiment of our framework, the maximum iteration number for the first and the second stages in the third step is 10 by default.

### 4.2 Results

Fig. 1 demonstrates how our K-core framework accelerates the label propagation algorithm [23] on com-Youtube and com-Amazon data. Because label propagation is very fast, we abandon the second stage algorithm for com-Amazon data. The plots on the left show the resulting modularity for different values of K. As K grows, the modularity for the com-Youtube data is generally constant, while the modularity for the com-Amazon data increases slightly. The black lines reflect that the proportion of remaining nodes and edges for increasing values of K, and the nodes reduce much faster than the edges as expected. Correspondingly as shown by red lines, the time to extract the K-core (Step 1) is negligible, and the running time for the main algorithm working on the K-core (Step 2) reduces almost proportionally to the node reduction. The time for labeling peripheral nodes outside the K-core (Step 3) grows very slowly starting from zero. The big reduction in time for the main algorithm and the small additional algorithmic cost result in significant time reduction for the whole algorithm.

Fig. 2 and Fig. 3 show the performance of our framework for five different main algorithms on twelve real world data sets, using the local modularity maximization as the second stage algorithm in step 3. Each subfigure in Fig. 2 represents the resulting modularity for each data set and each in Fig. 3 represents the normalized total running time. For social network data, from Fig. 2a to Fig. 4a we can see that for stage 2 in step 3, the local modularity maximization performs slightly better than the label propagation, but by Fig. 3a and 3b we can see that the running time is similar. The best choice usually depends on the algorithm, the data set, and the quality measurement.

In short, our framework can reduce the running time more than 80% with quality preserved for the Louvain method [1], Martelot’s method [16], modularity optimization [21], and Bayesian method [12] on most of the graphs. Such property holds for spectral clustering [32] on collaboration networks. Our framework for label propagation [23] works well on relatively large graphs, and can achieve a even better quality within a shorter time.

### 5. THEORETICAL ANALYSIS

The correlation coefficient measures the degree similarity of nodes connected by the same edges. When it is positive (assortative), nodes of high degree tend to connect to other high degree nodes, while if it is negative (dissortative), they tend to connect to low degree nodes [20]. In this section, we assume the correlation coefficient of the graphs is zero for simplicity.

#### 5.1 Stochastic Blockmodels and Variable Definitions

In stochastic blockmodels, communities are specified. Nodes within a community are randomly connected with a high probability, while nodes from different communities are loosely
connected. When inferring the communities for a given network without any other information, a likelihood function can be used to evaluate how likely the graph is generated by a graph model with such a community configuration [12].

Recently, Karrer and Newman proposed a generalized stochastic blockmodel [15] that can describe graphs with broad degree distributions. In their analysis, for a given graph $G$ with a specific community configuration $g$, they first define

$$m_{rs} = \sum_{ij} A_{ij} \delta(g_i, r)\delta(g_j, s),$$

(3)

where $g$ is a vector indicating the community assignment for each node, $A$ is the adjacency matrix of $G$, and $\delta$ is the Kronecker delta function.

Then the likelihood function of is as follows:

$$L(G|g) = \sum_{r,s} m_{rs} \log \frac{m_{rs}}{\kappa_r \kappa_s},$$

(4)

where $\kappa_r = \sum_s m_{rs}$ is the sum of degrees in community $r$. We also define the total edges of the graph as $m = \frac{1}{2} \sum_{r,s} m_{rs}$. We use $G_K$ to represent the $K$-core subgraph, and $g_K$ as the community labels for the nodes in the subgraph.

5.2 Likelihood of a Community Configuration in the Full Graph and the K-core Subgraph

In this section, we use the stochastic blockmodel above and an assumption defined as follows. Let $c$ be the number of communities. To analyze, first we categorize the edges into $\binom{c}{2} + c$ groups, where each edge in a group connecting nodes with a specific pair of community labels. For example, an edge in group $(r, s)$ means the two nodes it connected belong to community $r$ and $s$ respectively. If we randomly select a node through a randomly picked edge in a group, we use $\hat{y}(k)$ to represent the degree distribution of the node. In this section, we assume all the groups have the same $\hat{y}(k)$, which we call the identical degree distribution assumption.

Intuitively, under the identical degree distribution assumption, groups of nodes with denser connections (intra-community connections) will still have dense connection in the $K$-core, and vice versa for inter-community connections. This property allows the community detection algorithms to find the correct community labels in the $K$-core, as they do in the whole graph.

In other words, if there is a community configuration $\hat{g} = \arg \max_g L(G|g)$ and $\hat{g}_K = \arg \max_{g_K} L(G|g_K)$, then for each node $i$ in the $K$-core subgraph, we have $g(i) = g_K(i)$, assuming node $i$ has the same index in vector $g$ and $g_K$.

In the following part of this section, we will prove this property in terms of the likelihood function given by Eq. (4).

Let $y(k)$ be the probability density function for the degrees of the nodes in the full graph. When picking a node through a randomly selected edge, the probability of the node degree is not only determined by the degree distribution, but is
Figure 2: Change of Modularity over Different $K$'s, Using Modularity Maximization in Stage 2 of Step 3
Figure 3: Change of Running Time over Different K’s, Using Modularity Maximization in Stage 2 of Step 3
also proportional to the degree of the node. Let \( \hat{y}(k) \) be the probability density function of such node degrees. We have
\[
\hat{y}(k) = \frac{k \times y(k)}{\sum_{k'=0}^{\max_k} [k' \times y(k')]}.
\] (5)

In this subsection we use \( R \) to represent the ratio of edges in the \( K \)-core. Because each edge connects two nodes, the expected ratio \( E_R \) is \( \left[ \sum_{k'=0}^{\max} \hat{y}(k) \right]^2 \) as the correlation coefficient is zero.

Without loss of generality, we consider a single group and use \( j \) to represent \( m_{rs} \) in that group for simplicity. Obviously, \( R \) follows this distribution
\[
Pr(R) = f(R \times j; j, E_R),
\] (6)
where \( f \) is the PDF of a binomial distribution. Consequently, the variance of \( R \) is
\[
\delta(R) = \frac{iE_R(1 - E_R)}{j^2} = \frac{E_R(1 - E_R)}{j}.
\] (7)

We also define a function \( h_j(R) \)
\[
h_j(R) = R \log R.
\] (8)

Let \( \theta \in (0, 1) \) be a small real number. Consequently, we have the following lemma.

**Lemma 5.1.** The expected value of \( h_j(R) \) can be approximated by
\[
E[h_j(R)] \approx E_R \log E_R + \frac{1 - E_R}{2j},
\] (9)
where the magnitude of the error is less than \( E_R \theta^3 / 6 = O(\theta^3) \) with probability
\[
1 - \frac{\delta_R^2}{\theta^2 E_R^2} = 1 - \frac{(1 - E_R)^2}{\theta^2 j^2}.
\] (10)

**Proof.** We start from the Taylor expansion of \( h_j(R) \) about \( R = E_R \):
\[
h_j(R) = E_R \log E_R + (\log E_R + 1)(R - E_R)
\]
\[+ \frac{1}{1 \times 2E_R} (R - E_R)^2 - \frac{1}{2 \times 3 E_R^2} (R - E_R)^3 + \cdots.
\] (11)

Because \( E[(R - E_R)^2] = \delta_R \), the expectation of \( h_j(R) \) is
\[
E[h_j(R)] = E_R \log E_R + \frac{1}{1 \times 2E_R} \frac{E_R(1 - E_R)}{j}
\]
\[- \frac{E_R}{2 \times 3} E[(R - E_R)^3] + \cdots.
\] (12)

where the \( \text{th} \) term \((i > 1) \) above is
\[
\frac{E_R}{(i - 1) \times i} E[(R - E_R)^i].
\]

Hence, if the magnitude of \( \frac{E_R}{E_R} - 1 \) is less than one, the summation of the remaining terms is less than the third term.

Assume \( \left| \frac{E_R}{R} - 1 \right| < \theta \), we use Chebyshev’s inequality to obtain the corresponding probability
\[
Pr(\left| \frac{E_R}{E_R} - 1 \right| < \theta) = Pr(\left| R - E_R \right| < \theta E_R)
\]
\[\geq 1 - \frac{\delta_R^2}{(\theta E_R)^2}
\]
\[= 1 - \frac{(1 - E_R)^2}{\theta^2 j^2}.
\] (13)

Consequently, if Eq. (12) is truncated from its third term, we can obtain Eq. (9). The truncation error is less than the third term and consequently less than \( E_R \theta^3 / 6 \) with a probability given by Eq. (13). \( \square \)

The error bound given by Lemma 5.1 is quite rigorous in practice. For example, we assume that there are at least 10 edges in each group, namely, \( j = 10 \), and about half of the edges are in the \( K \)-core, so \( E_R = 0.5 \). We want to maintain
a high probability, i.e., $Pr(\frac{1}{E_R} - 1 < \theta) > 0.95$, then we have
\[
\theta \leq \sqrt{\frac{(1 - 0.5)^2}{(1 - 0.95) \times 10^2}} \approx 0.22. \tag{14}
\]

This equation means the summation of the remaining terms $O(\frac{1}{E_R} - 1)^3$ in Eq. (12) is less than $E_R\theta^6/6 = 8.9 \times 10^{-4}$ with high probability. Consequently, the first term in Eq. (12) is at least 170 times larger than the residue in magnitude.

In the following theorem, we use $g^{(1)}$ and $g^{(2)}$ as two different community configurations. Let $g^{(1)}_K$ and $g^{(2)}_K$ as the community labels for nodes in the subgraph, and for each node $i$ we define $g^{(1)}_K(i) = g^{(1)}(i)$ and $g^{(2)}_K(i) = g^{(2)}(i)$. The likelihood function is defined by Eq. (4).

**Theorem 5.2.** Under the identical degree distribution assumption, if $G$ is a graph with correlation coefficient equal to zero, for two different community configurations $g^{(1)}$ and $g^{(2)}$, without loss of generality, we assume
\[
\mathcal{L}(G|g^{(1)}) \geq \mathcal{L}(G|g^{(2)}).
\tag{15}
\]

Then for a subgraph $G_K$ of $G$, the corresponding $g^{(1)}_K$ and $g^{(2)}_K$ are expected to have the following relationship with a large probability:
\[
\mathcal{L}(G_K|g^{(1)}_K) \geq \mathcal{L}(G_K|g^{(2)}_K).
\tag{16}
\]

**Proof.** We use $R_{rs}$ to represent the an instance of $R$ corresponding to $m_{rs}$. According to the assumption, all the $R_{rs}$ have the same expected value $E_R$. Because the total intra-community degree is typically large, the variation of the edge reduction is very small, and therefore $\kappa$ is expected to reduce to $E_{RR_{rs}}$. Then the likelihood for the subgraph is
\[
\mathcal{L}(G_K|g_K) = \sum_{r,s} R_{rs} m_{rs} \log \frac{R_{rs} m_{rs}}{E_R \kappa_{rs}}
= \sum_{r,s} [R_{rs} m_{rs} \log \frac{m_{rs}}{\kappa_{rs}} + R_{rs} m_{rs} \log E_R]
- R_{rs} m_{rs} \log E^2_R.
\tag{17}
\]

Consequently, the expected likelihood is
\[
E[\mathcal{L}(G_K|g_K)] = E_R \sum_{r,s} m_{rs} \log \frac{m_{rs}}{\kappa_{rs}} + \sum_{r,s} m_{rs} E[R_{rs} \log R_{rs}]
= E_R \mathcal{L}(G|g) + \sum_{r,s} m_{rs} E[h_{m_{rs}}(R_{rs})]
- 2mE_R \log E_R
\approx E_R \mathcal{L}(G|g) + \sum_{r,s} \frac{1 - E_R}{2} - 2mE_R \log E_R.
\tag{18}
\]

where the approximation is by Lemma 5.1 and $m = \frac{1}{2} \sum_{r,s} m_{rs}$ is used.

Once $G$ is given, $m$ is determined; and once $G_K$ is given, $E_R$ is determined. Hence, we can see that when all the other parameters fixed, $\mathcal{L}(G_K|g_K)$ is expected to grow as $\mathcal{L}(G|g)$ grows. Then, we can reach the conclusion of the theorem.

**Theorem 5.2.** Justifies the idea that, if the algorithm is to maximize the likelihood function, the community labels of the remaining nodes given $G_K$ is expected to be the same as when given $G$.

In addition, because the degree correlation will only affect the remaining ratio and will not affect the correctness of Lemma 5.1, Theorem 5.2 also holds for assortative or dissortative graphs.

**6. CONCLUSIONS**

In this work, we introduce a $K$-core based framework that can accelerate community detection algorithms significantly. It includes three steps: first, find the $K$-core of the original graph; second, run the community detection algorithm on the $K$-core; third, find labels for all the nodes outside the $K$-core and then optimize for the whole graph. Experiments demonstrate efficiency and accuracy on different real graphs, under the measurement of modularity.

Theoretical analysis supports our approach through $K$-core: if the main algorithm is to maximize the likelihood of a stochastic blockmodel, for nodes in $K$-core, the community configuration inferred from the $K$-core is expected to be the same as it is inferred from the whole graph. This means the step two in our framework is a good initial guess for community detection of the whole graph in step three, but much faster because the problem size is smaller.

The parameter $K$ needs to be determined according to different algorithms or data sets. We can choose a $K$ to meet the following two requirement for the $K$-core subgraph: the ratio of nodes should be small enough to gain a sufficient acceleration, while the ratio of edges should not be too small due to the approximation in the proof of Theorem 5.2. Both requirements depend on the type of graphs, and in practice, we suggest the node remaining ratio to be between 10% to 50% and the edge remaining ratio to around 50%.

In short, with an appropriate $K$, our framework can reduce the running time of community detection algorithms while preserving the detection quality.

**7. ACKNOWLEDGMENTS**

The authors would like to thank Prof. David E. Keyes for his important help to complete this project. The work of the first author was supported by the King Abdullah University of Science and Technology (KAUST) baseline research fund and the KAUST Strategic Initiative for Extreme Simulation. The work of the second and third authors was funded by the GRAPHIS Program at DARPA. Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-AC04-94AL85000.
8. REFERENCES

[1] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[2] Z. Bu, C. Zhang, Z. Xia, and J. Wang. A fast parallel modularity optimization algorithm (fpmqa) for community detection in online social network. *Knowledge-Based Systems*, 50:246–259, 2013.

[3] A. Clauset, M. E. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70(6):066111, 2004.

[4] S. N. Dorogovtsev, A. V. Goltsev, and J. F. F. Mendes. K-core organization of complex networks. *Physical Review Letters*, 96(4):040601, 2006.

[5] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[6] S. Fortunato and C. Castellano. Community structure in graphs. In *Computational Complexity*, pages 490–512. Springer, 2012.

[7] U. Gargi, W. Lu, V. S. Mirrokni, and S. Yoon. Large-scale community detection on youtube for topic discovery and exploration. In *Proceedings of the 5th International AAAI Conference on Weblogs and Social Media*, 2011.

[8] R. Ghosh and K. Lerman. Community detection using a measure of global influence. In *Advances in Social Network Mining and Analysis*, pages 20–35. Springer, 2010.

[9] C. Giatsidis, D. M. Thilikos, and M. Vazirgiannis. Evaluating cooperation in communities with the k-core structure. In the *IEEE International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 87–93, 2011.

[10] M. Girvan and M. E. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[11] G. Guimerà, S. Mossa, A. Turtschi, and L. N. Amaral. The worldwide air transportation network: Anomalous centrality, community structure, and cities' global roles. *Proceedings of the National Academy of Sciences*, 102(22):7794–7799, 2005.

[12] J. M. Hofman and C. H. Wiggins. Bayesian approach to network modularity. *Physical Review Letters*, 100(25):258701, 2008.

[13] J. Huang, H. Sun, Y. Liu, Q. Song, and T. Weninger. Towards online multiresolution community detection in large-scale networks. *PloS one*, 6(8):e23829, 2011.

[14] J. Janson and M. J. Luczak. A simple solution to the k-core problem. *Random Structures & Algorithms*, 30(1-2):50–62, 2007.

[15] B. Karrer and M. E. Newman. Stochastic blockmodels and community structure in networks. *Physical Review E*, 83(1):016107, 2011.

[16] E. Le Martelot and C. Hankin. Fast multi-scale detection of relevant communities in large-scale networks. *The Computer Journal*, 2013.

[17] J. Leskovec, K. J. Lang, and M. Mahoney. Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th International Conference on World Wide Web*, pages 631–640. ACM, 2010.

[18] F. Malliaros, V. Megalooikonomou, and C. Faloutsos. Fast robustness estimation in large social graphs: Communities and anomaly detection. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, volume 12, pages 942–953, 2012.

[19] M. Newman. Spectral methods for network community detection and graph partitioning. arXiv:1307.7729, 2013.

[20] M. E. Newman. Assortative mixing in networks. *Physical Review Letters*, 89(20):208701, 2002.

[21] M. E. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6):066133, 2004.

[22] M. E. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113, 2004.

[23] U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3):036106, 2007.

[24] J. Riedy, D. A. Bader, and H. Meyerhenke. Scalable multi-threaded community detection in social networks. In the *26th IEEE International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW)*, pages 1619–1628. IEEE, 2012.

[25] O. Riordan. The k-core and branching processes. *Combinatorics, Probability & Computing*, 17(1):111, 2008.

[26] Y. Ruan, D. Fuhry, and S. Parthasarathy. Efficient community detection in large networks using content and links. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 1089–1098, 2013.

[27] M. Salathé and J. H. Jones. Dynamics and control of diseases in networks with community structure. *PLoS Computational Biology*, 6(4):e1000736, 2010.

[28] A. E. Sarıyüce, B. Gedik, G. Jacques-Silva, K.-L. Wu, and U. V. Catalyurek. Streaming algorithms for k-core decomposition. *Proceedings of the VLDB Endowment*, 6(5), 2013.

[29] M. Á. Serrano and M. Boguñá. Clustering in complex networks. *Physical Review Letters*, 96(4):040601, 2006.

[30] A. Clauset, M. E. Newmann, and C. Moore. Finding and evaluating community structure in networks. *Physical Review E*, 70(6):066111, 2004.

[31] C. Giatsidis, D. M. Thilikos, and M. Vazirgiannis. Evaluating cooperation in communities with the k-core structure. In the *IEEE International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 98–102, 2011.

[32] U. Von Luxburg. A tutorial on spectral clustering. *Journal of Statistical Mechanics: Theory and Experiment*, 2007(10):P10008, 2008.

[33] Z. Zhu, C. Wang, L. Ma, Y. Pan, and Z. Ding. Scalable community discovery of large networks. In *the 9th International Conference on Web-Age Information Management*, 2008 (WAIM’08), pages 381–388, 2008.