Article

Coreness Variation Rule and Fast Updating Algorithm for Dynamic Networks

Liang Gao 1,*, Ge Gao 1, Dandan Ma 1 and Lida Xu 2,*

1 Key Laboratory of Transport Industry of Big Data Application Technologies for Comprehensive Transport, Beijing Jiaotong University, Beijing 100044, China; 16120803@bjtu.edu.cn (G.G.); 13251045@bjtu.edu.cn (D.M.)
2 College of Life Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China
* Correspondence: lianggao@bjtu.edu.cn (L.G.); xuld@mail.buct.edu.cn (L.X.)

Received: 17 March 2019; Accepted: 1 April 2019; Published: 3 April 2019

Abstract: Coreness is one of the important indicators to measure the importance of a node. Traditionally, the coreness of a node is measured by $k$-core decomposition. However, to measure the coreness in a dynamic network, the $k$-core decomposition method becomes very time-consuming and inefficient, and cannot meet the need in very large real networks. Recently, the $H$ operator method was proposed to calculate the coreness of a node, which provides a novel method to deal with the coreness of a node in a network. In this paper, we decode the coreness variation rule by a symmetric pair of experiments, i.e., deleting and adding edge, on real networks. Then, an algorithm to fast update the coreness of related nodes is proposed. Results on five real networks showed that the performance of the proposed algorithm was greatly enhanced and comprehensively superior to the $k$-core decomposition algorithm. Our study provides a promising way to optimize the algorithm of coreness calculation in the dynamic networks.

Keywords: dynamic network; coreness; $H$ operator; coreness updating algorithm

1. Introduction

Identifying critical nodes accurately is of great importance in order to understand the structure and function of a network [1–5]. This will allow us to better control many kinds of dynamical processes, such as the outbreak of epidemics [6–8], conduct successful advertisements for e-commercial products [9], and prevent catastrophic outages in power grids [10]. The widely used metrics to measure the node importance are degree, closeness [11], betweenness [12], H-index [13], coreness [14] and so on. It is generally believed that nodes that have more neighbors are more important than those having less neighbors [15–17]. Recently, Kitsak et al. [18] argued that the location of a node is more significant than the number of its linked neighbors. Many research results show that coreness is a better indicator of a node’s influence on spreading dynamics than degree [18–22].

The traditional way of calculating the coreness of a node is the $k$-core decomposition [23,24], as shown in Figure 1. In a simple undirected network, all the isolated nodes form the 0-shell are removed from the network. Next, it removes all nodes with degree $k \leq 1$. This causes new nodes with degree $k = 1$ to appear. These nodes are also removed, and the process is continued until the nodes remaining are those of degree $k > 1$. The removed nodes and their associated links form the 1-shell. The pruning process is repeated to remove the nodes with degree $k \leq 2$ iteratively, and all the nodes and links removed in this round constitute the 2-shell. The process is continued until all higher-layer shells have been identified and all nodes have been removed. Then, each node $i$ is assigned a shell layer $c_i$, called the coreness of node $i$. The $k$-core decomposition is simple and intuitive process used to calculate the coreness of nodes. However, when considering a dynamic network whose topological structure is
varying, the $k$-core decomposition process has to be restarted in order to determine the coreness of nodes, which will be a time-consuming task, especially for a large-scale dynamic network. Therefore, it is of great necessity to establish a fast updating method to handle the coreness determination in a dynamic network [24–27].

![k-core decomposition](image)

**Figure 1.** $k$-core decomposition.

Lv, et al. [28] proposed an asynchronous updating algorithm to calculate the $k$-shell indices. In this algorithm, they defined an operator $H$, which reveals the relationship between degree, H-index and coreness and obtains a new method for calculating the coreness of nodes. In each step, it randomly selects one node to update its intermediate value towards the $k$-shell index by the operator $H$, the degree of each node is used as the initial input of the operator $H$. This operator $H$ is iteratively calculated until the value for each node goes into its steady state. Ultimately the steady value of each node is its coreness. Compared with the traditional $k$-core decomposition, this algorithm uses only local information of the networks and can implement a distributed computing, which significantly improves the computational efficiency. Subsequently, Lee et al. [29] proposed two methods to optimize the asynchronous iterative order of nodes, which further improved the computational efficiency of the asynchronous updating algorithm.

From the $k$-core decomposition method [23,24] to the asynchronous updating algorithm [28], then to the algorithm that optimizes the asynchronous iteration order of nodes [29], people have made great progress in the coreness calculation.

In this paper, we study how the coreness of nodes will vary in a dynamic network with local structural variation when the initial coreness of each node is known. Mastering the variation rules of the coreness will not only provide a theoretical basis to improve the computational efficiency of coreness calculation, but also provide a computing basis for the study related to the application of the coreness in the dynamic networks.

The rest of this paper is organized as follows. In Section 2, we introduce $H$ operator method in detail. Section 3 is devoted to reveal the coreness variation rule by a symmetric pair of experiments on real networks. The CHDE and CHAE algorithms are proposed to fast update the coreness during the local structural variation as well. Results on five real networks are shown in Section 4. Section 5 provides our conclusions.

### 2. $H$ Operator Method for Coreness Calculation

Denote $G(V, E)$ as a simple network, where $V$ is the set of nodes and $E$ is the set of links. The degree and coreness of an arbitrary node $i \in V$ are denoted as $k_i$ and $c_i$, respectively. The number of nodes and links are represented by $|V| = N$ and $|E| = M$, respectively.
According to the research of Lv et al. [24], when the $\mathcal{H}$ operator reaches the steady state, the stable value of each node is the coreness. At this time, the coreness of node $i$ and the coreness of its neighbors $j_1, j_2, \ldots, j_k$ still satisfy the relation of operator $\mathcal{H}$, that is:

$$c_i = \mathcal{H}\left(c_{j_1}, c_{j_2}, \ldots, c_{j_k}\right).$$

(1)

The operator $\mathcal{H}$ relationship is shown in Figure 2. Assuming $j \in \Gamma_i = \{j_1, j_2, \ldots, j_k\}$, where $\Gamma_i$ is the set of the neighbor nodes of node $i$. All the neighbors of node $i$ are arranged along the horizontal axis by a decreasing order of their coreness, $c_j$. Then the coreness of node $i$ is equal to the side length of the largest square with no points inside (Figure 2), i.e., $c_i = h$.

![Figure 2. Illustration of the coreness calculation process via $\mathcal{H}$ operator method.](image)

### 3. Coreness Variation in Dynamic Networks

In this section, we will discuss how the coreness of nodes will vary when there is a local variation in the network structure. It is necessary to clarify the time scale of variations before discussing the variations of the network structure. The structure of some networks varies very slowly. For example, infrastructure networks such as road networks, railway networks and so on. Their structural variations are counted by year. The structure of some networks varies rapidly. For example, WWW, instant messaging networks, social media networks, etc. Their structural variations are counted by second. When the time scale of the observation is approximately equivalent to the time scale of the structural variation, the observed variations in the network structure can be considered as local variations. This paper interprets the local variations in the network structure as: “deleting an edge” or “adding an edge” which are denoted as DE experiment and AE experiment, respectively.

We executed DE and AE experiments on five real networks. Their basic topological characteristics are shown in Table 1. We tried to answer the following questions: (1) which nodes will get their coreness varied? (2) What conditions are met by the nodes that their coreness will vary?

| Network             | $N$  | $M$  | $\bar{K}$ | $L$    | $C$    |
|---------------------|------|------|-----------|--------|--------|
| Email               | 1133 | 5450 | 9.622     | 3.606  | 0.220  |
| US Power Grid       | 4941 | 6594 | 2.669     | 18.990 | 0.080  |
| Petster-Hamster     | 2426 | 16631| 13.711    | 3.588  | 0.538  |
| ca-AstroPh          | 18,771| 198,050| 21.102    | 4.194  | 0.631  |
| loc-brightkite      | 58,228| 214,078| 7.035     | 3.789  | 0.172  |

#### 3.1. DE Experiments

Denote $e_{ij} = (i, j)$ as the deleted edge in the network, where $i$ and $j$ represent the nodes forming the edge. Results show that the nodes whose coreness has varied in DE experiment have the same coreness before the edge deleted, i.e., they all come from the same $k$-shell layer, and fall into the same $(k-1)$-shell layer.
3.1.1. Classification of Nodes with the Same Coreness

Assuming that the neighbor nodes of node \( i \) are arranged in the descending order by their coreness as \( c_{j(1)} \geq c_{j(2)} \geq \cdots \geq c_{j(k_1-1)} \geq c_{j(k_1)} \). Comparing \( c_{j(h+1)} \) and \( h \), all nodes with the coreness \( h \) can be divided into \( T_1 \) and \( T_2 \) types as the following conditions.

\( T_1: \ c_{j(h)} \geq h > c_{j(h+1)} \). Which means, the coreness of the first \( h \) neighbor nodes is greater than, or equal to \( h \) (green zone in Figure 3a), and the coreness of rest neighbor nodes is less than \( h \) (red zone in Figure 3a).

\( T_2: \ c_{j(h)} \geq h = c_{j(h+1)} \). Which means, the coreness of the first \( h \) neighbor nodes is greater than or equal to \( h \) (green zone in Figure 3b), the coreness of the \( (h+1) \) th neighbor node is \( h \) (point A of Figure 3b), and the coreness of rest neighbor nodes is less than or equal to \( h \) (red zone in Figure 3b).

The classification above is a complete classification of the nodes having the same coreness \( h \), that is, \( T_1 + T_2 \) will contain all the nodes with the same coreness \( h \).

3.1.2. Coreness Variation Rule and CHDE Algorithm

When \( c_i \neq c_j \), we assume \( c_i (= h) < c_j \). Node \( j \) just loses a neighbor in the red zone in Figure 2, and its coreness does not vary. While, node \( i \) loses a neighbor in the green zone in Figure 2, and whether its coreness varies will be determined by the following coreness variation rules.

(1) If \( i \in T_1 \), then \( c_i^{DE} = c_i - 1 = h - 1 \). Here, \( c_i^{DE} \) represents the coreness of node \( i \) after the edge \( e_{ij} \) deleted. If the coreness variation begins from node \( i \), and it can spread out, then the coreness variation can only spread in the connected subgraph, \( V_i(h) \), which includes node \( i \), and in which all nodes have the same initial coreness \( h \).

(2) If \( i \in T_2 \), then \( c_i^{DE} = c_i = h \), and the coreness of other nodes will not vary.

According to the coreness variation rules mentioned above, combined with the \( H \) operator method [29], we propose the CHDE algorithm to quickly update the coreness of related nodes. The process is as follows:

**Step 1:** Judge the type of node \( i \). If \( i \in T_1 \), then find out the connected subgraph \( V_i(h) \) by the searching algorithm (Figure 4). Otherwise if \( i \in T_2 \), the coreness of any node will not vary, and the CHDE algorithm ends.

**Step 2:** Update the coreness of all nodes in the connected subgraph \( V_i(h) \) by the \( H \) operator method [29].
When \( c_i = c_j \), the coreness variations of node \( i \), node \( j \), and other related nodes can still be determined and updated by the CHDE algorithm. The modification in ‘Step 1’ of CHDE algorithm is to judge the type of both node \( i \) and node \( j \), and find out the connected subgraph \( V_i(h) \) and \( V_j(h) \). The modification in ‘Step 2’ is updating the coreness in both \( V_i(h) \) and \( V_j(h) \).

### 3.2. AE Experiments

Denote \( e_{ij} = (i, j) \) as the added edge in the network, where \( i \) and \( j \) represent the nodes forming the edge. Results show that the nodes whose coreness varies in the AE experiment have the same coreness before the edge added, i.e., they all come from the same \( k \)-shell layer, and go up into the same \((k+1)\)-shell layer.

#### 3.2.1. Classification of Nodes with the Same Coreness

Assuming that the neighbor nodes of node \( i \) are arranged in the descending order by their coreness as \( c_{j(1)} \geq c_{j(2)} \geq \cdots \geq c_{j(k-1)} \geq c_{j(k)} \). Comparing \( c_{j(h)} \) and \( h \), all nodes with the coreness \( h \) can be divided into \( T3 \) and \( T4 \) types as the following conditions.

**T3:** \( c_{j(h)} \geq h + 1 > h \geq c_{j(h+1)} \). Which means, the coreness of the first \( h \) neighbor nodes is greater than or equal to \( h + 1 \) (green zone in Figure 5a), and the coreness of rest neighbor nodes is less than \( h \) (red zone in Figure 5a).

![Figure 5. Coreness distribution in the neighborhood of (a) T3 and (b) T4 nodes.](image)

**T4:** \( c_{j(h-1)} \geq h = c_{j(h)} \geq c_{j(h+1)} \). Which means, the coreness of the first \( h - 1 \) neighbor nodes is greater than or equal to \( h \) (green zone in Figure 5b), the coreness of the \( h \)-th neighbor node is \( h \) (point B in Figure 5b), and the coreness of rest neighbor nodes is less than or equal to \( h \) (red zone in Figure 5b).

The classification above is a complete classification of the nodes having the same coreness \( h \), that is, \( T3 + T4 \) will contain all the nodes with the same coreness \( h \).
3.2.2. Coreness Variation Rule and CHAE Algorithm

When \( c_i \neq c_j \), we assume \( c_i (= h) < c_j \). Node \( j \) gets a new neighbor in the red zone in Figure 2, and its coreness does not vary. While, node \( i \) gets a neighbor in the green zone in Figure 2, and whether its coreness varies will be determined by the following coreness variation rules.

1. If \( i \in T_3 \), then \( c_i^{AE} = c_i + 1 = h + 1 \). Here, \( c_i^{AE} \) represents the coreness of node \( i \) after the edge \( e_{ij} \) added. Particularly, the coreness of other nodes in the network does not vary.

2. If \( i \in T_4 \), whether the \( c_i^{AE} \) increase to \( h + 1 \) cannot be determined directly. Then, a searching algorithm has to be employed find out the connected subgraph \( V_i(h) \), which includes node \( i \), and in which all nodes have the same initial coreness \( h \).

According to the coreness variation rule mentioned above, combined with the \( \mathcal{H} \) operator method [29], we propose the CHAE algorithm to quickly update the coreness of related nodes. The specific process is as follows:

Step 1: Judge the type of node \( i \). If \( i \in T_4 \), then find out the connected subgraph \( V_i(h) \) by the searching algorithm (Figure 4). Otherwise, if \( i \in T_3 \), \( c_i^{AE} = c_i + 1 = h + 1 \), then the CHAE algorithm ends.

Step 2: Set the coreness of all nodes in the connected subgraph \( V_i(h) \) as \( h + 1 \), then update the coreness of all nodes in \( V_i(h) \) by the \( \mathcal{H} \) operator method [29].

When \( c_i = c_j \), the coreness variations can be determined and updated by the CHAE algorithm. Since node \( i \) and node \( j \) are connected together, the CHAE algorithm can be applied directly without any modification.

4. Case Study

In this section, we define the relative computational efficiency to test the computational efficiency of CHDE and CHAE algorithms, with respect to the \( k \)-core decomposition algorithm. The relative computational efficiency is tested on the five real networks (Table 1). There are five combinations of the type of node \( i \) and node \( j \) in DE and AE experiments (Table 2). It is worth noting that when \( c_i = c_j \), the node \( i \) and node \( j \) are completely equivalent in DE and AE experiments, so the two combinations \((i \in T_2 \text{ and } j \in T_1 \text{ in DE}) \text{ and } (i \in T_4 \text{ and } j \in T_3 \text{ in AE})\) are not listed in Table 2. We will not discuss the combinations (2) and (5) in DE and combinations (1) and (3) in AE in the rest part of this section either, since our method is obviously better than \( k \)-core decomposition method. Because the time consumption of our method is almost zero in these combinations, while the \( k \)-core decomposition method still has to decompose the network.

**Table 2.** Five combinations of the type of node \( i \) and node \( j \) in DE and AE experiments.

|       | \( c_i < c_j \) | \( c_i = c_j \) |
|-------|-----------------|-----------------|
| **DE** | (1) \( i \in T_1 \) | (3) \( i \in T_1 \text{ and } j \in T_1 \) | (4) \( i \in T_1 \text{ and } j \in T_2 \) | (5) \( i \in T_2 \text{ and } j \in T_2 \) |
| **AE** | (1) \( i \in T_3 \) | (2) \( i \in T_4 \) | (3) \( i \in T_3 \text{ and } j \in T_3 \) | (4) \( i \in T_3 \text{ and } j \in T_4 \) | (5) \( i \in T_4 \text{ and } j \in T_4 \) |

4.1. Computational Efficiency of CHDE Algorithm

For each combination, 100 edges are randomly selected to be deleted for each real network. Then, both CHDE and \( k \)-core decomposition algorithms are applied to calculate the coreness of nodes, and the time consumption is recorded as \( t_{H,DE}(c_{ij}) \) and \( t_{kcore,DE} \), respectively. Due to the randomness in the searching process of CHDE algorithm and the decomposing process of \( k \)-core decomposition algorithm, therefore, when an edge is deleted, the coreness of nodes is calculated 100 times independently for both algorithms. To prove the improvement of the computational efficiency...
of the CHDE algorithm quantitatively comparing to the $k$-core decomposition algorithm, we define the relative computational efficiency:

$$R_{H,DE} = \frac{\min_{m=1..100} t^m_{k-core,DE}(e_{ij})}{\max_{m=1..100} t^m_{H,DE}(e_{ij})}$$

where $m$ represents the $m$th calculation when $e_{ij}$ is deleted. In the definition, we let the shortest time consumed in 100 calculations (best) of the $k$-core decomposition algorithm compare to the longest time consumed in the 100 calculations (worst) of the CHDE algorithm when $e_{ij}$ deleted.

Results (Table 3) show that the computational efficiency of CHDE algorithm is generally better than the $k$-core decomposition algorithm. All results in Table 3 are obviously larger than 1, which means CHDE algorithm is overall more efficient than $k$-core decomposition algorithm. On average, the lowest computational efficiency of CHDE algorithm is still 20.30 times of $k$-core decomposition algorithm, which appears on the US Power Grid network. Among the BEST group results, CHDE algorithm achieves more than 1000 times more efficient than $k$-core decomposition algorithm in two combinations related to ca-AstroPh and loc-brightkite networks. Generally speaking, the larger the network is, the more remarkable the advantage of CHDE algorithm is.

**Table 3. Relative computational efficiency ($R_{H,DE}$) of CHDE Algorithm for five real networks.**

| Real Network       | $c_i < c_j$ | $c_i = c_j$ | $c_i > c_j$ |
|--------------------|-------------|-------------|-------------|
|                    | $i \in T_1$ | $i \in T_1$ | $i \in T_1$ |
|                    | $j \in T_1$ | $j \in T_1$ | $j \in T_2$ |
| Email              | 426.98      | 301.58      | 281.19      |
|                    | 200.89      | 50.80       | 31.73       |
| US Power Grid      | 474.56      | 245.87      | 379.19      |
|                    | 285.67      | 20.30       | 60.02       |
| Petster-Hamster    | 772.38      | 801.65      | 697.01      |
|                    | 409.58      | 229.86      | 145.83      |
| ca-AstroPh         | 1185.59     | 1139.01     | 1079.72     |
|                    | 648.10      | 251.57      | 186.91      |
| loc-brightkite     | 986.95      | 1125.96     | 1066.01     |
|                    | 526.01      | 685.42      | 416.63      |

### 4.2. Computational Efficiency of CHAE Algorithm

For each combination, 100 pairs of nodes are randomly selected to be added for each real network. Then, both CHAE and $k$-core decomposition algorithms are applied to calculate the coreness of nodes, and the time consumption is recorded as $t_{H,AE}(e_{ij})$ and $t_{kcore,AE}$, respectively. Due to the randomness in the searching process of CHAE algorithm and the decomposing process of $k$-core decomposition algorithm, in the instance of when an edge is added, the coreness of nodes is therefore calculated 100 times independently for both algorithms. To prove the improvement of the computational efficiency of the CHAE algorithm quantitatively comparing to the $k$-core decomposition algorithm, we define the relative computational efficiency:

$$R_{H,AE} = \frac{\min_{m=1..100} t^m_{k-core,AE}(e_{ij})}{\max_{m=1..100} t^m_{H,AE}(e_{ij})}$$
where \( m \) represents the \( m \)-th calculation when the edge \( e_{ij} \) is added. In the definition, we let the shortest time consumed in 100 calculations (best) of the \( k \)-core decomposition algorithm compare to the longest time consumed in the 100 calculations (worst) of the CHAE algorithm when \( e_{ij} \) added.

Results (Table 4) show that the computational efficiency of CHAE algorithm is generally better than the \( k \)-core decomposition algorithm. Similar to CHDE, all results for CHAE algorithm in Table 4 are obviously larger than 1 as well, which proves CHAE algorithm is totally more efficient than \( k \)-core decomposition algorithm. On average, the lowest computational efficiency of CHAE algorithm is still 8.93 times of \( k \)-core decomposition algorithm, which appears on the US Power Grid network, too. In the worst situation, \( R_{H,AE} \) is smaller than \( R_{H,DE} \). The possible reason is that the connected subgraph \( V_i(h) \) for AE (adding an edge) is more likely to be larger than that for DE (deleting an edge). Among the BEST group results, CHAE algorithm achieves more than 1000 times more efficient than \( k \)-core decomposition algorithm in all three combinations on ca-AstroPh network. Similar to CHDE algorithm, the larger the network is, the more remarkable the advantage of CHAE algorithm is.

| Real Network        | \( c_i < c_j \) | \( c_i = c_j \) | \( c_i > c_j \) |
|---------------------|------------------|-----------------|-----------------|
|                     | \( i \in T_4 \)  | \( j \in T_4 \) | \( j \in T_4 \) |
| Email               |                  |                 |                 |
| BEST                | 157.18           | 238.72          | 227.78          |
| AVERAGE             | 9.28             | 74.15           | 54.71           |
| US Power Grid       |                  |                 |                 |
| BEST                | 291.71           | 312.42          | 285.64          |
| AVERAGE             | 8.93             | 80.33           | 56.70           |
| Petster-Hamster     |                  |                 |                 |
| BEST                | 874.83           | 547.72          | 512.52          |
| AVERAGE             | 104.43           | 158.35          | 114.82          |
| ca-AstroPh          |                  |                 |                 |
| BEST                | 1018.30          | 1055.64         | 1059.94         |
| AVERAGE             | 170.80           | 233.28          | 229.48          |
| loc-brightkite      |                  |                 |                 |
| BEST                | 821.53           | 955.76          | 857.46          |
| AVERAGE             | 271.96           | 332.20          | 304.58          |

5. Discussion

When observing a dynamic network in a short enough time window, all the structural variations can be looked on as the local variation like deleting an edge or adding an edge. Based on this thinking, in this paper, we carried out DE and AE experiments to reveal the coreness variation rules, and presented CHDE and CHAE fast coreness updating algorithm.

In the DE and AE experiments, we found the following coreness variation rules: If a local structural variation can cause coreness variation of node(s), then all the nodes whose coreness varied will have the same initial coreness \( h \), i.e., the coreness before the DE and AE action. This means that all the nodes whose coreness varied, belonged to the same shell layer \( (h-\text{shell}) \). In DE experiments, these nodes’ coreness would decrease by 1, fall into the \( (h-1) \)-shell layer, and form one or two connected subgraph(s). In AE experiments, those nodes’ coreness would increase by 1, go up into the \( (h+1) \)-shell layer, and form one connected subgraph.

To reveal the mechanism of the coreness variation, all nodes with the same coreness (e.g., \( h \)) were divided into \( T_1 \) and \( T_2 \) types in DE experiments (Figure 3), and \( T_3 \) and \( T_4 \) types in AE experiments (Figure 5). In DE experiment, the variation of coreness was caused by \( T_1 \) node of the deleted edge. Beginning the searching process from the \( T_1 \) node will find out one or two connected subgraph(s) with the initial coreness \( h \), which is the outmost range of the nodes whose coreness will vary. Comparing to the DE experiment, the situation in AE experiment was a little bit more complex. The reliable strategy is
to begin the searching process from the $T_4$ node of the added edge, and find out a connected subgraph with the initial coreness $h$, which is the outmost range of the nodes whose coreness could vary.

According to the experimental results mentioned above, we established the CHDE and CHAE algorithms to fast updating the coreness for the local structural variation of deleting and adding an edge. Results showed that the proposed algorithms are overall superior to the $k$-core decomposition algorithm in terms of computational efficiency.

This paper is a start on the exploration of how the coreness of nodes varies, and how to fast update the coreness in a tolerable time when there is a local variation in the network structure. There are still many problems to be discussed in future. For instance, how the coreness of nodes will vary when more complex structural variations happen; how to go a further step to optimize the computational efficiency of the coreness updating algorithm; etc.

6. Conclusions

In this paper, we first revealed the coreness variation rule by a symmetric pair of experiments, i.e., DE and AE experiments. Then, CHDE and CHAE algorithms were proposed to fast update the coreness during the local structural variation of a dynamic network. Results on five real networks proved that the performance of the proposed method was greatly enhanced compared with the $k$-core decomposition method.

Author Contributions: Conceptualization, L.G. and L.D.; methodology, L.G. and L.D.; formal analysis, L.G., G.G. and L.D.; investigation, L.G. and G.G.; validation, L.G. and G.G.; writing—original draft preparation, L.G., G.G. and L.D.; writing—review and editing, L.G., G.G., D.M. and L.D.

Funding: This research was funded by the National Natural Science Foundation of China (No.91646124, No.71571017, No.71621001, and No.91746201).

Acknowledgments: The authors thank for support from the Fundamental Research Funds for the Central Universities (2018JBM026).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Yu, S.; Gao, L.; Xu, L.; Gao, Z.Y. Identifying influential spreaders based on indirect spreading in neighborhood. *Phys. A* 2019, 523, 418–425. [CrossRef]
2. Gao, L.; Shan, X.; Qin, Y.; Yu, S.; Xu, L.; Gao, Z.Y. Scaling tunable network model to reproduce the density-driven superlinear relation. *Chaos* 2018, 28, 033122. [PubMed]
3. Gao, L.; Miao, Y.; Qin, Y.; Zhao, X.; Gao, Z.Y. Finding topological center of a geographic space via road network. *Phys. A* 2015, 419, 128–133. [CrossRef]
4. Zeng, G.; Li, D.; Guo, S.; Gao, L.; Gao, Z.; Stanley, H.E.; Havlin, S. Switch between critical percolation modes in city traffic dynamics. *Proc. Natl. Acad. Sci. USA* 2019, 116, 23–28. [CrossRef]
5. Wang, P.; Lü, J.; Yu, X. Identification of Important Nodes in Directed Biological Networks: A Network Motif Approach. *PLoS ONE* 2014, 9, e106132. [CrossRef] [PubMed]
6. Pastor-Satorras, R.; Vespignani, A. Immunization of complex networks. *Phys. Rev. E* 2002, 65, 036104. [CrossRef]
7. Cohen, R.; Benavraham, D.; Havlin, S. Efficient immunization of populations and computers. *Phys. Rev. Lett.* 2002, 91, 12343.
8. Pastor-Satorras, R.; Castellano, C.; Van Mieghem, P.; Vespignani, A. Epidemic processes in complex networks. *Rev. Mod. Phys.* 2015, 87, 925–979. [CrossRef]
9. Leskovec, J.; Adamic, L.; Huberman, B. The dynamics of viral marketing. *ACM Trans. Web.* 2007, 1, 5. [CrossRef]
10. Albert, R.; Albert, I.; Nakarado, G.L. Structural vulnerability of the North American power grid. *Phys. Rev. E* 2004, 69, 025103. [CrossRef]
11. Bavelas, A. Communication patterns in task-oriented groups. *J. Acoust. Soc. Am.* 1950, 22, 725–730. [CrossRef]
12. Freeman, L. Centrality in social networks: Conceptual clarification. *Soc. Netw.* 1979, 1, 215–239. [CrossRef]
13. Hirsch, J. An index to quantify an individual’s scientific research output. *Proc. Natl. Acad. Sci. USA* 2005, 102, 16569–16572. [CrossRef] [PubMed]

14. Dorogovtsev, S.; Goltsev, A.; Mendes, J. K-core organization of complex networks. *Phys. Rev. Lett.* 2006, 96, 040601. [CrossRef]

15. Albert, R.; Jeong, H.; Barabasi, A.-L. Error and attack tolerance of complex networks. *Nature* 2000, 406, 378–482. [CrossRef]

16. Pastor-Satorras, R.; Vespignani, A. Epidemic spreading in scale-free networks. *Phys. Rev. Lett.* 2001, 86, 3200–3203. [CrossRef] [PubMed]

17. Cohen, R.; Erez, K.; ben-Avraham, D.; Havlin, S. Breakdown of the Internet under intentional attack. *Phys. Rev. Lett.* 2001, 86, 3682–3685. [CrossRef]

18. Kitsak, M.; Gallos, L.K.; Havlin, S.; Liljeros, F.; Muchnik, L.; Stanley, H.E.; Makse, H.A. Identification of influential spreaders in complex networks. *Nat. Phys.* 2010, 6, 888–893. [CrossRef]

19. Pei, S.; Muchnik, L.; Andrade, J.S., Jr.; Zheng, Z.; Makse, H.A. Searching for super spreaders of information in real-world social media. *Sci. Rep.* 2014, 4, 5547. [CrossRef]

20. Castellano, C.; Pastor-Satorras, R. Competing activation mechanisms in epidemics on networks. *Sci. Rep.* 2012, 2, 371. [CrossRef] [PubMed]

21. Liu, Y.; Tang, M.; Zhou, T.; Do, Y. Core-like groups result in invalidation of identifying super-spreader by k-shell decomposition. *Sci. Rep.* 2015, 5, 9602. [CrossRef] [PubMed]

22. Hu, Q.; Gao, Y.; Ma, P.; Yin, Y.; Zhang, Y.; Xing, C. A new approach to identify influential spreaders in complex networks. *Acta Phys. Sin.* 2013, 62, 140101.

23. Seidman, S.B. Network structure and minimum degree. *Soc. Netw.* 1983, 5, 269–287. [CrossRef]

24. Batagelj, V.; Zaversnik, M. An O(m) algorithm for cores decomposition of networks. *arXiv* 2003, arXiv:cs/0310049.

25. Sarıyüce, A.E.; Gedik, B.; Jacques-Silva, G.; Wu, K.L.; Çatalyürek, Ü.V. Incremental k-core decomposition: Algorithms and evaluation. *VLDB J.* 2016, 25, 425–447. [CrossRef]

26. Jin, H.; Wang, N.; Yu, D.; Hua, Q.S.; Shi, X.; Xie, X. Core Maintenance in Dynamic Graphs: A Parallel Approach Based on Matching. *IEEE Trans. Parallel Distrib. Syst.* 2018, 29, 2416–2428. [CrossRef]

27. Wen, D.; Qin, L.; Zhang, Y.; Lin, X.; Yu, J.X. I/O Efficient Core Graph Decomposition: Application to Degeneracy Ordering. *IEEE Trans. Knowl. Data Eng.* 2019, 31, 75–90. [CrossRef]

28. Lü, L.; Zhou, T.; Zhang, Q.; Stanley, H.E. The h-index of a network node and its relation to degree and coreness. *Nat. Commun.* 2016, 7, 10168. [CrossRef]

29. Lee, Y.; Zhou, T. Fast asynchronous updating algorithms for k-shell indices. *Phys. A* 2017, 482, 524–531. [CrossRef]