Large-Scale Social Network Privacy Protection Method for Protecting K-Core

Jian Li[1], Xiaolin Zhang[1], Jiao Liu[1], Lu Gao[1], Huanxiang Zhang[2], and Yueyang Feng[3]
(Corresponding author: Xiaolin Zhang)
School of Information Engineering, Inner Mongolia University of Science and Technology[1]
Baotou 014010, China
School of Computer Engineering and Science, Shanghai University[2]
Shanghai 200000, China
School of Science, Inner Mongolia University of Technology[3]
Hohhot 010010 , China
Email: fyylijian@outlook.com
(Received Feb. 3, 2020; Revised and Accepted Sept. 7, 2020; First Online May 30, 2021)

Abstract

Social network analysis has many important applications and methods which depend on the sharing and publishing of graphs. For example, link privacy requires limiting the probability of an adversary identifying a target sensitive link between two individuals in the published social network graph. However, the existing link privacy protection methods have low processing power for large-scale graph data and less consideration of community protection in the publishing graphs. Therefore, aiming at sensitive link privacy protection, a large-scale social network privacy protection model to protect K-Core (PPMPK) was proposed. The large-scale social network graph was processed to ensure that the core number and the community structure of the nodes were unchanged based on the Pregel parallel graph processing model. Extensive experiments on the real data sets showed that the proposed method could effectively process the large-scale graph data and protect the data availability of the published graphs, especially in community protection.

Keywords: Community Protection; K-Core; Pregel; Privacy Protection; Social Network

1 Introduction

With the development of the social network, the Internet plays an increasingly important role in daily life. Online Social Network saves a large number of users' personal information. For example,the facebook's daily active users reached 1.401 billion in 2017 and generated PB-level data every hour. These information contains both sensitive and nonsensitive data. The large-scale social network graph data are saved in the cloud environment which provide a trusted mobile terminal and a secure environment [7, 8] before user privacy sensitive data are anonymity. How to protect the privacy of sensitive data has attracted more and more researchers' attention. The researchers Mining the information in the social network can discover the hidden structure of the network. Community structure is an important feature and widespread in the real-world social network.

Typically, a community is a collection of nodes in the network that have the same or similar roles [9, 12]. However, people who use social network applications are always facing serious privacy breaches and malicious attacks. Each element that makes up a social network may lead to the disclosure of user privacy information. The main privacy of social network includes node attribute privacy and edge privacy. At present, the study on social network privacy protection methods has made positive progress. Researchers have proposed various privacy protection methods for different social network privacy issues. Node attribute privacy includes node identification or sensitive attributes such as name, phone number, address, etc. Researchers used data generalization [20], perturbation [13], or adding noise nodes [15] to protect node attribute privacy.

For edge privacy, researchers proposed different privacy protection models to protect the edge between nodes according to different attack backgrounds. Random perturbation technique, which randomly modifies the original data to reduce the attacker's inferred confidence is able to resist complex background attacks. In this paper, we decompose social network graph to get the node core number. The information interchange between nodes based on Pregel model. The edges are simultaneously perturbed according to the probability while ensured that the core number of nodes are unchanged in the social network graph. Our main contributions can be summarized as follows:

1) Aiming at dealing with large-scale social network graphs, we propose a search result of reachable nodes (SRRN) algorithm that quickly finds the neighbor nodes' information based on the Pregel model.

2) According to the neighbor information, we propose a large-scale social network privacy protection model (PPMPK) for protection of K-Core. The K-Core decomposition graph is used to measure the influence of the node. The edges is perturbed in the social network while protecting the core number of the nodes.

3) Rigorous experiments have been performed on five real world social network data sets. The promising experimental result has demonstrated that the proposed approach can achieve comparably good results when compared with the previous privacy protection approaches. These results also verify the efficacy of protecting the community structure.

The rest of the paper is organized as follows. Section 2 reviews the related works on protecting social networks' privacy and Section 3 details the define preliminaries. Section 4 proposes the privacy protection model. The specific algorithm steps are presented in Section 5. The experimental evaluation is performed in Section 6. Section 7 concludes the paper and points out the future work.

2 Relate Works

At present, most of the edge privacy protection methods adopt anonymous model to prevent the leakage of private information and malicious attacks. It is an efficient anonymous model to protect sensitive edge based on random perturbation technology. In Reference [16], the random probability model was used to implement privacy protection by adding, deleting and switching edges. This method caused the perturbed graph to be a random graph or even a useless graph. In order to improve the efficiency of anonymous graphs, Zhang [19] proposed the perturbation based on the subgraph structure. This method divided the original network graph into several subgraphs and randomly perturbed the edge in subgraphs which increased the degree of some nodes. The probability that these nodes are identified in the subgraph was increased. To solve this problem, a formula that balances the degree of nodes was applied to reduce the probability [5]. Ford [6] proposed a local neighbor random perturbation algorithm running on a single workstation. This method randomly selected the local neighbors of the source node to replace the target node to protect the sensitive edges. In Reference [20], a distributed random perturbation algorithm was designed to solve the problem of large-scale social networks. Furthermore, a sensitive area random perturbation algorithm was provided to improve data availability [17].

Community structure is an important feature of social network graph. Anonymous social networks while protecting the community structure has become a hot topic. The spectrum of the graph serves as an important topological feature of the graph. Ying and Wu [14] compared the similarity between nodes, randomly adding and deleting k edges. This method protected the spectrum of graph changing to protect the community structure. Campan [4] used the community partition algorithm based on graph segmentation theory to added and deleted edges by calculating the Laplacian matrix. Zheng [21] used differential privacy for privacy protection to the online social network structure query, and protected the social network community structure while anonymization. Kumar [9] used the upper approximation concept of rough sets to divide the community for anonymity, but the algorithm execution efficiency is low. Zhang [18] provided a method to anonymited the dynamic social network while protected the community structure as possible.

In summary, the random perturbation algorithm can protect the edge privacy well. Anonymous methods for protecting community structure based on community division have a problem of low processing efficiency for large-scale social network graph. Therefore, a new solution is proposed. After random perturbance anonymity, the influence of the node is maintained, then the stability of the community is guaranteed. The k-core decomposition graph is used to measure the influence of the node. Seidman [12] first proposed the concept of k-core and gave the k-core decomposition process. Kitsak [1] studied the relationship between the k-core and the speed of disease propagation in the social network. Experiments showed that the k-core has more reliable results than the node degree and betweenness centrality. In Reference [11], a privacy protection algorithm maintained the core number of nodes unchanged by the operation of adding, deleting, and switching edges on single workstation. To this end, the proposed privacy protection model PPMPK differs from the traditional numerical perturbations or graph modification methods in that it paralleling selects neighbor nodes for perturbation. At last, the result protected the community's stability under the conditions of privacy.

3 Related Definitions

Let G = (V, E) denotes a directed social graph, consisting of a set of vertices V and a set of edges E. $V = \{v_1, v_2, \cdots, v_n\}$, each of these nodes corresponds to a real user in the social network. $E_{ij} =< v_i, v_j >$ indicates a directed social edge by user i to user j, where i is the source node and j is the destination node.

Definition 1. *(R-neighbor) r is a given non-negative integer. node u, v are two different node in the graph. Dist(u, v) represents the shortest path length of node u to node v. if node v satisfies the condition, node v is said to be the r-neighbor of node u:*

$$Dist(u, v) \leq r \tag{1}$$

Definition 2. *(K-Core) Let k be an integer. The degree of any node v of the set C∈V is not less than k, and*

the largest subgraph G_c (C, E |C) derived therefrom is called K-Core. The largest subgraph that obtained after recursively removing the nodes with degrees less than k and the edges expected to be connected is the K-Core.

Definition 3. *(Core number) If node v belongs to the k-core and does not belong to the $(k+1)$-core, the core number of node v is k. The k-core decomposition is to decompose the social network graph into the largest subgraph from the border of the social network to the central. Each node in the social network graph will have a core number after decomposition.*

Definition 4. *(K-shell) A group of nodes with the core number of k is called a k-shell.*

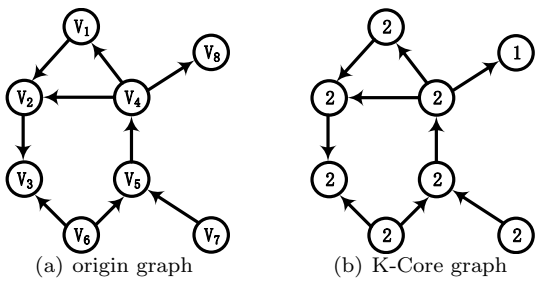
Definition 5. *(Ef_degree) In the k-core decomposition process, the number of the high core or the same code nodes connected is called node effective degree.*

Definition 6. *(K-lamina) In the K-Core decomposition process, nodes with higher effective degree than k in the same k-shell are called k-lamina type nodes.*

Definition 7. *(K-corona) In the k-core decomposition process, nodes with the equal number of effective degree and core number in the same k-shell are called k-corona type node.*

4 Large-Scale Social Network Privacy Protection Model for Protecting K-Core

In the social network, if you want to hide the existed edge <u, v>, you only need to hide the source node u or the destination node v. Only the source or destination node is known, the existence of the edge cannot be inferred [6]. Therefore, a large-scale social network privacy protection model PPMPK for protecting k-cores is proposed to anonymous social network. The main idea is to preserve the edge $< u, v >$ with a probability $p(0 \le p \le 1)$. If the probability of the edge <u, v> is $(1-p)$, the algorithm replaces each of the node with node w. The privacy level of the published graph can be adjusted according to the probability p. In order to get a privacy protection model, some examples and properties are given first.



(a) origin graph (b) K-Core graph

Figure 1: Origin graph and K-Core decomposition graph

Example 1. *Figure 1(a) is the original social network. Figure 1(b) is k-core decomposition graph with node label which is the core number of nodes. As shown in Figure 1(a), it is assumed that the edge $<v_4, v_8>$ is deleted. v_4 is 2-corona node and v_8 is 1-corona type node. As defined by the k-core, the decomposition graph first decomposes the shell of the lower core number nodes. Node v_8 and connected edges are first decomposed. The deleted edges have no effect on node v_4 because v_4 is a higher core number compared to the node v_8. As defined by k-corona, the core number of k-corona nodes is equal to the number of ef_degree, so the core number of node v_8 is downgraded.*

Property 1. *The edge that a higher core number node connected a lower core number node is deleted. The K-Core of the higher core number node is not affected. In lower core number nodes, the core number of k-corona type nodes is downgraded, but the k-lamina type nodes is not downgraded.*

Example 2. *As shown in Figure 1(a), it is assumed that the edge$<v_2$,$v_3>$ is deleted. v_2 is 2-corona and v_3 is 2-lamina. As defined by k-lamina and k-corona, the ef_degree of k-lamina type node is higher than core number. if the edge $<v_2$,$v_3>$ is deleted, the core number of node v_2 will unchange. The core number of v_3 will downgraded.*

Property 2. *The edge that a higher core number node connected a lower core number node is deleted. The k-core of the higher core number node is not affected. In lower core number nodes, the core number of k-corona type node is downgraded, but the k-lamina type node is not downgraded.*

Example 3. *As shown in Figure 1(a), it is assumed that the edge $<v_6, v_3>$ is deleted. v_6 and v_3 are k-corona type nodes with same core number. According to the k-corona definition, the ef_degree of the node is equal to the core number. If the edge $<v_6, v_3>$ is deleted, the core number of nodes v_6 and v_3 are both downgraded.*

Property 3. *The edge of k-corona type node connection with the same-core is deleted. The core number of nodes are both downgraded.*

Lemma 1. *(K-Core perturbance safety condition) For each edge, it is divided into 3 conditions by node type. If only $\forall <u,v> \in E$ satisfies the following conditions while perturbing the edge by choosing candidate node w, the core number of nodes does not change.*

1) $\forall$core[u]>core[v], $\forall$w$\in$V:core[w]$\geq$core[v]$\wedge$(w,v)$\notin$E;
2) $\forall$core[u]=core[v]$\wedge$ef_degree[u]>core[u],$\forall$w$\in$V: core[w]$\geq$core[v]$\wedge$(w,v)$\notin$E;
3) $\forall$core[u]=core[v]$\wedge$ef_degree[u]=core[u]$\wedge$ef_degree[v] =core[v],$\forall$w,q$\in$V:core[w]$\geq$core[v]$\wedge$(w,v)$\notin$E, core[q]$\geq$core[u]$\wedge$(q,u)$\notin$E;

Definition 8. *In the anonymous process of the directed graph G={ V, E}, the K-Core perturbance safety conditions (KPSC) is the sufficient condition for implementing the random perturbation model for protecting the K-Core.*

The KPSC condition 1, (∀core[u]>core[v]) indicates that the edge connection type is the higher core number node connected to the lower core number node. (∀w∈V: core[w]≥core[v]∧(w, v)∉E) shows the candidate node w which is the neighbor of node u and not connected to node v, then edge <u, v> is deleted and <w, v> is added. For high core number nodes u and w, deleting and adding edges that connected lower core number has no effect on the nodes. For the lower core number node v, deleting and adding a high-core number edge, the core number of nodes has no effect.

The KPSC Condition 2,(∀core[u]=core[v]∧ef_degree[u] >core[u]) indicates that the edge type is connected to the same core node. there exists a k-lamina type node. (∀w∈V: core[w]≥core[v]∧(w,v)∉E) expresses the candidate node w which is the higher core number neighbor of node u and is not connected to node v.

- If the *k*-lamina type nodes connected to the k-corona type node, the neighbor of k-corona type node is selected as a candidate. The KPSC condition 3 proves that the core number of perturbed k-corona type nodes has no effect.

- If the *k*-lamina type nodes connected to the k-lamina type nodes. the total number of edges are unchanged in the perturbation. In the same k-shell, the k-lamina type nodes with a smaller number of connected edges are preferentially decomposed. That is, the k-lamina type nodes with added edges will be decomposed, so adding edges has no effect on the core number of nodes.

The KPSC condition 3, (∀core[u]=core[v]∧ef_degree[u] =core[u]∧ef_degree[v]=core[v]) indicates that the edge is connected to the same-core k-corona type node. (∀w, q∈V: core[w]≥core[v]∧(w, v)∉E, core[q]≥core[u]∧ (q, u)∉E) indicates the condition for the selection of candidate nodes. For the k-corona type node with the same core number, ef_degree[v]=core[v] means that if an edge is deleted, the core number will downgraded, so the perturbance is twice. The nodes connected to the same core number are in the same shell. The k-core decomposition first decomposes the k-corona type node and the edge connected to it, so deleting and adding the connection edge has no effect on the k-corona type candidate node.

Based on the above, the KPSC condition can protect the core number of nodes while perturbing the three type edges in the social network graph, so the goal of protecting the k-core must satisfy the KPSC in the process of the perturbance. The KPSC is implemented based on a sufficient condition for protecting the large-scale social network privacy protection model of the k-core. A large-scale social network privacy protection model that protects the k-core for a given directed graph G=$\{V, E\}$, the perturbation neighbor r, the probability p. The main steps are:

1) The algorithm decomposes the social network graph to get the node core number and finds the r reachable neighbor of the node.

2) The probability is assigned to the edge in the graph. If the assignment probability is p, the edge is preserved. If the assignment probability is 1-p, the edge is perturbed. The perturbance edge is determined according to the edge retention probability. The candidate nodes are selected according to the node core number. The candidate nodes selection process satisfies the KPSC.

5 Large-Scale Social Network Privacy Protection Algorithm for Protecting K-Core

In order to obtain the candidate nodes that meet the random perturbation of directed graph. A node reachability search K-Core algorithm is proposed. The algorithm performs a reachable list. Each node generates a random neighbor table (RNT). As shown in Table 1, the RNT consists of (srcid, dstid, hops, core, ef_degree) and each row of the RNT list is an Random Neighborhood Table Entry (RNTE), Table 1 is the initialization RNT list in Figure 1 (such as 11022 indicates that the node v_1 is initialized with a core number of 2 and the effective degree of 2). In the RNTE:

- Source node (srcid): Source node id;
- Destination node (disid): When the node is initialized, the label value is the source node id. During the information transmission, the label value is the source node reachable destination node id.
- Hops: The number of hops indicates that the node needs several iterations to pass information to the destination node;
- Core number: The number of k-core of the nodes;
- Effective degree(ef_degree): The number of effective degrees of the node.

5.1 Node Reachability Search Algorithm based on K-core

The Search results of reachable nodes (SRRN) algorithm is based on the Pregel model. The Pregel model searches for reachable nodes through message transmission, message merging, and message processing functions. At last, SRRN generates a list of nodes RNTs.We present the pseudo-code of SRRN processes in Algorithm 1.

The SRRN algorithm initializes the RNT list of the node to determine whether the node is an active node

Table 1: Initialize the node RNT table

Node	RNTE
V_1	11022
V_2	22023
V_3	33022
V_4	44023
V_5	55023
V_6	66022
V_7	77011
V_8	88011

Algorithm 1 Search Results of Reachable Nodes (SRRN)

Input: Social network graph G, reachable parameter r
Output: Node reachable list

```
1: Initialization: RNT (G);
2: SuperStep=0;
3: while SuperStep ≤ r do
4:   for v_i∈V∩v_i is active do
5:     for Message from vi.srcRNT do
6:       Merge(Message);
7:     end for
8:     if Message.IsNotExist(v_i.Attr) then
9:       Update RNT.hop+1;
10:      Update RNT.srcid=vi.id;
11:    else
12:      VoteToHalt();
13:    end if
14:    if New AddMessage in Message then
15:      Send New AddMessage to vi.disid;
16:    end if
17:  end for
18: end while
```

according to whether the node has out degree. If the node status is inactive, the node don't need to calculate. If the node status is active, the update messages insert according to the messages are not exist in the RNT list and send the updated messages to the destination node.

5.2 Perturbation Algorithm that Keeps the Core Number of Nodes

Algorithm 2 shows the core preserving perturbation of node (CPPN) algorithm. The CPPN algorithm obtains the forward and back RNT list information of the node through the SRRN algorithm. The CPPN algorithm uses the random function to determine whether to perform edge perturbation. If the assignment probability is p, the edge is preserved. If the assignment probability is 1-p, the random perturbation edge of the candidate node is selected. The edge connection condition is determined by the RNTE value of the node. Finally, the random edge perturbation is performed according to different situations.We present the pseudo-code of SRRN processes in Algorithm 2.

Algorithm 2 Perturbation Algorithm for core preserving perturbation of node (CPPN)

Input: Social network graph G, perturbance probability P
Output: Perturbation edge result list

```
1: Initialization: SRRN(G); SRRN(G.reverse);
2: ResultList=∅;
3: for (u,v)∈ E do
4:   if Random==P then
5:     ResultList← (u,v);
6:   else
7:     if core[u]>core[v] or core[u]<core[v] then
8:       High_To_Lower(u,v);
9:     end if
10:    if ef_degree[u]==ef_degree[v]∩ef_degree[u]
       ==core[u] then
11:      Corona_To_Corona(u,v);
12:    else
13:      Lamina_To_Lamina(u,v);
14:    end if
15:  end if
16: end for
17: return ResultList;
```

It is judged whether the edge is perturbed based on the probability function. If the perturbation is required, the edge is perturbed according to the different connecting situation. If the high-core node is connected to the low-core node, and the algorithm 3 is selected. The algorithm 4 is selected when the k-corona node with the same-core is connected. If the K-lamina type node with the same core number is existed in the connected edge , the algorithm 5 is selected. Algorithm 3 is a perturbation edge algorithm for high-core connected low-core. Lines 3-7 and 18-22 ensure that the core number of nodes is unchanged and the reachability between nodes is guaranteed. Lines 8 and 23 indicate that if no candidate nodes can guarantee the reachability between the nodes, then the candidate nodes in the opposite direction are selected to ensure that the core number is unchanged.

Figure 2 shows the results of perturbing the edges that high-core nodes connect to low-core. Table 2 shows the RNT list of nodes (only the RNTE value of the perturbed nodes are listed, the bolds are the back propagation RNTE value, and the RNTE (1) represents the 1-neighbor of node). If perturbing the edge $<v_4, v_8>$, the algorithm compares the core number of nodes v_4 and v_8 in the RNT list. The result indicates that the connected edge is a high-core connected low-core. Deleting the connected edge $<v_4, v_8>$, the algorithm selects the reachable neighbor of the high-core node v_4. The candidate nodes are $\{v_5, v_6, v_7, v_1, v_2, v_3\}$, and the nodes $\{v_1, v_2, v_3\}$ are preferentially selected. The preferentially selected nodes ensure the core number of nodes unchanged while ensuring the reachability between nodes. Finally, the algorithm randomly selects the node v_1, and adds an

Algorithm 3 High-core connection low-core perturbance (High_To_Lower)

Input: An edge <u,v>
Output: The perturbed edge result

```
1: Initialization: candidate=∅; Array=∅;
2: if core[u]>core[v] then
3:    for u.reverse(RNT) do
4:       if core[candidate]≥core[v] and candidate isNo-
         tExist in v.RNT then
5:          Add candidate to Array;
6:       end if
7:    end for
8:    if Array.Size==0 then
9:       for u.RNT do
10:         if core[candidate] ≥core[v] and candidate is-
            NotExist in v.RNT then
11:            Add candidate to Array;
12:         end if
13:      end for
14:   end if
15:   return ResultList←(Random(Array),v);
16: else
17:   for v.RNT do
18:      if core[candidate] ≥core[u] and candidate isNo-
         tExist in v.RNT then
19:         Add candidate to Array;
20:      end if
21:   end for
22:   if Array.Size==0 then
23:      for v.reverse(RNT) do
24:         if core[candidate] ≥ core[u] and candidate is-
            NotExist in v.RNT then
25:            Add candidate to Array;
26:         end if
27:      end for
28:   end if
29:   return ResultList←(u,Random(Array));
30: end if
```

edge$<v_1, v_8>$; For the perturbed edge $<v_7, v_5>$, the algorithm compares the core number of nodes v_7 and v_5 in the RNT list. The RNT list indicates that the connected edge type is a low-core connected high-core. Finally, the algorithm deletes the connected edge $<v_7, v_5>$, and selects the neighbor of the high-core node v_5. The candidate nodes have $\{v_6, v_4, v_1, v_2, v_8\}$. The algorithm randomly selects node v_6, and adds edges $<v_7, v_6>$.

Algorithm 4 is k-corona type nodes perturbation algorithm. Lines 2-8 are the first perturbation, and ensure that the core number of node v does not change. Lines 10-16 are the second perturbation, and ensure that the core number of node u does not decrease.

Figure 3 shows the result of the connected edge perturbation of the k-corona node. Table 3 shows the RNT list of the perturbed nodes, the perturbed edge is $<v_6, v_3>$. The core number and effective degree of nodes v_6 and v_3 are compared in the RNT list. The result shows that

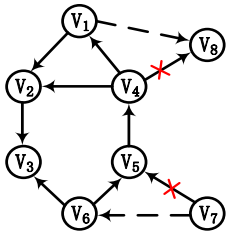


Figure 2: High-core connection low-core perturbance results

Table 2: The RNT list of nodes

Node	RNTE(1)	RNTE(2)
V_4	45122	46222
	41122	47211
	42123	**43222**
	48111	
V_8	**84123**	**85222**
V_5	56122	**52223**
	54123	**51222**
	57111	**58222**
V_7	75122	74223

Algorithm 4 K-Corona nodes perturbation (Corona_To_Corona)

Input: An edge <u,v>
Output: The perturbed edge result

```
1: Initialization: candidate=∅; Array=∅;
2: for u.RNT do
3:    if core[candidate] ≥core[u] and candidate isNotEx-
      ist in v.RNT then
4:       Add candidate to Array;
5:    end if
6: end for
7: ResultList ← (Random(Array),v);
8: candidate=null; Array=null;
9: for v.RNT do
10:   if core[candidate] ≥core[v] and candidate isNotEx-
      ist in v.RNT then
11:      Add RNT.disid to Array;
12:   end if
13: end for
14: ResultList←(u,Random(Array));
15: return ResultList;
```

the edge with same-core and needs twice perturbations. The first perturbation ensures that the core number of nodes v_6 is unchanged. The perturbance selects the high-core neighbors of the node v_3 are $\{v_1, v_2, v_4\}$. The result preferentially selects the node v_2, adds the connection edge $<v_6, v_2>$. The second perturbation guarantees the core number of node v_3 and selects the high-core neighbor v_5, v_4 of node v_6, and prefer to select node v_5. The result adds an edge $<v_5, v_3>$.

Table 3: The RNT list of the perturbed nodes

Node	RNTE(1)	RNTE(2)
V_6	**63122**	**64223**
	65122	
V_3	36122	34223
	32123	31222

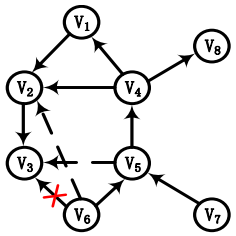


Figure 3: Perturbance result of the same-core K-Corona type node

Algorithm 5 is k-lamina type node perturbation algorithm. The two nodes have at least one k-lamina type. The algorithm first selects the high-core or same-core neighbors of the k-lamina type node. If the neighbor of the k-lamina type node is always selected, the k-lamina type node will become the k-corona type node. At last, the core number of node will downgrade. The anonymous program will update the ef_degree value when run at lines 5-10.

Algorithm 5 Same-core k-lamina node perturbance (Lamina_To_Lamina)

Input: An edge <u,v>
Output: The perturbed edge result ResultList

```
1: if core[u] < ef_degree then
2:    Select u.RNT;
3:    ResultList← (Random(Array),v);
4:    Update(u.ef_degree-1);
5: end if
6: if core[v]< ef_degree then
7:    Select v.RNT;
8:    ResultList← (u,Random(Array));
9:    Update(u.ef_degree-1);
10: end if
11: return ResultList;
```

Figure 4 shows the results of the perturbation of the k-lamina type nodes. Table 4 shows the RNT list of the perturbed nodes. The perturbed edge $<v_4, v_1>$ is required. The core number and the effective degree of nodes v_4 and node v_1 are compared in the RNT list. It shows that nodes v_4 and node v_1 are the k-lamina type node with the same-core. The neighbor of the node v_4 is selected. The candidate nodes have $\{v_3, v_5, v_6\}$. The node v_3 is randomly selected, and the ef_degree of the node v_4 is updated.

Table 4: The results of the perturbation of the k-lamina type nodes

Node	RNTE(1)	RNTE(2)
V_4	45422	46222
	41122	47211
	42123	**43222**
V_1	14123	15222
	12123	**13222**

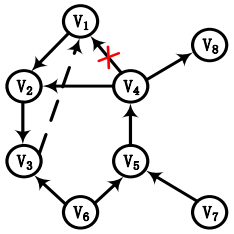


Figure 4: The same core K-lamina type node perturbance results

5.3 Algorithm Security Analysis

PPMPK obtains an anonymous social network graph through random perturbations. G*={V*, E*} is the anonymous result graph obtained by deleting and adding edges under the edge retention probability p of the original social network graph G={V, E}. If the edge assignment probability is 1-p, the edge is deleted. The candidate node is added. Assuming that all edges are perturbed once. Even if there are k-corona-type nodes connected, the two perturbations will increase the security of the connected edges. We suppose the number of nodes and edges do not changed, $|V| = |V^*|$,$|E| = |E^*|$, and $|E|$ is the set of edges in the original social network graph G. If the edge retention probability is 0, the social network graph is transformed into a completely random graph, so that the data availability of the published graph is zero.

In the random perturbation graph, the attacker wants to identify the target node, and must judge the possibility of the connected edge according to the edge retention probability p and the anonymous graph G*. If it is a full graph random perturbance, the candidates for the target node x are all nodes from u∈G* (except v), because adding and deleting edges are completely random. The attacker wants to determine the exact probability of each edge as:

$$P_{r(p)} = \frac{1}{\begin{pmatrix} |E| \\ \frac{|E|}{1-p} \end{pmatrix} \begin{pmatrix} |\overline{E}| \\ \frac{|\overline{E}|}{1-p} \end{pmatrix}} \tag{2}$$

The denominator represents the probability of a random perturbation under the retention probability p. For PPMPK, the candidate node is restricted to the r neighbors. The probability that the attacker can recognize the

edge is proportional to Pr(p) and r value. By increasing the r value and retention probability p, the publisher can effectively prevent the attacker from identifying the real destination node.

6 Performance and Evaluation

The PPMPK privacy protection algorithm performs performance analysis and evaluation. The result compares PPMPK with R-SW and R-A/D algorithms [16], and uses different dimensions and methods for verification.

6.1 Experimental Setup

The algorithms were tested using five real social network datasets:

1) Karate Karate Club Network.
2) Jazz Jazz Musician Network.
3) URV email network.
4) Polblogs political blog data set.
5) Amazon dataset.

The datasets description is shown in Table 5. n is the number of nodes and m is the number of edges. $\overline{deg}$ represents the average degree and d represents the diameter.

Table 5: Data set

Dataset	n	m	$\overline{deg}$	d
Karate	34	78	4.588	5
Jazz	198	2742	27.697	6
URV email	1133	5451	9.622	8
Polblogs	1224	16715	27.312	8
Amazon	403394	2443408	12.114	15

The experimental operating environment: 15 computing nodes, CPU 1.8GHz, 16GB RAM, Hadoop 2.7.2, Spark2.2.0, programming language: Scala 2.11.12.

6.2 Running Time Analysis

Figure 5 is a comparison of PPMPK with R-A/D and R-SW (r = 3) for running time. It can be seen that PPMPK selects 5% to 25% of the edges for anonymity in the 3-hop neighbors of node in the Amazon dataset. When running the anonymous algorithm, the time differences between the R-SW and R-A/D is small. This is because the random algorithm does not need to calculate the candidate nodes, but the PPMPK algorithm needs to decompose the social network to obtaining a K-Core graph and requires information propagation between nodes when selecting candidate nodes, so PPMPK algorithm takes slightly longer than R-SW and R-A/D.

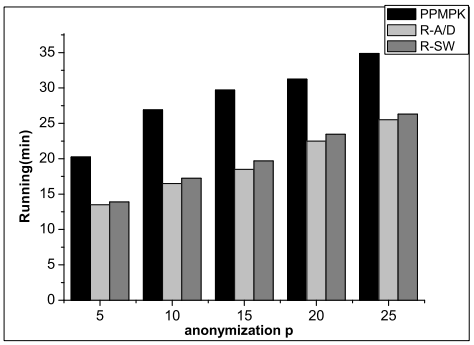


Figure 5: Running time comparison graph

6.3 Information Loss Analysis

We use $RelativeError = |u - u^*|/u$ (u and u^* refer to the metrics in the original graph and the perturbation graph) to evaluate the average shortest path and closeness centrality after the social network graph perturbance. The smaller the value, the smaller the information loss, the better the data availability is maintained.

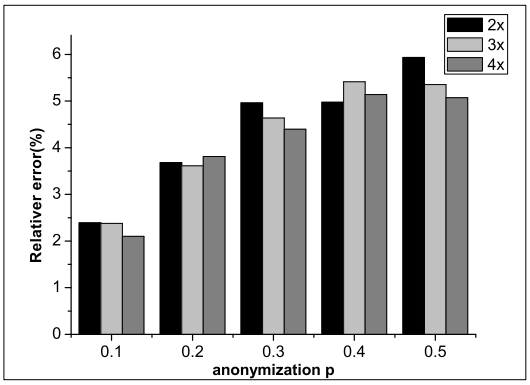


Figure 6: The relative error of average shortest path

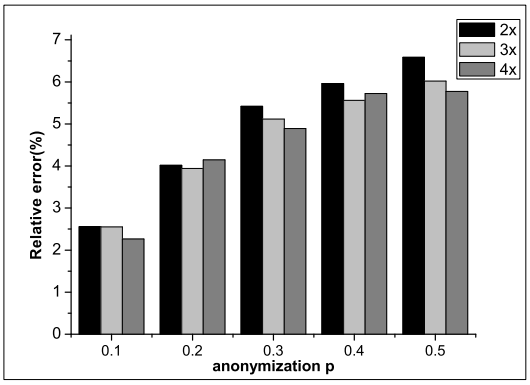


Figure 7: The relative error of closeness centrality

Figures 6 and 7 show different local neighbor r on the average shortest path and closeness centrality in the Polblogs dataset (Note: X-axis represents the percentage of perturbed edges, and Y-axis represents the relative error

rate after the perturbation). As the number of perturbed edges increases, the relative error of the average shortest path and the closeness center are increased. As the perturbation edges increase, the information loss is greater. When the number of perturbed edges is constant, the relative error may decrease as the perturbance r increases because the candidate node is far away from the source node. When adding edges,fewer vertices are changed in the path. In contrast, the distance between the candidate nodes and the source node are greatly change.

In order to measure the influence of the anonymous algorithm on the graph structure, the topological properties of the graph are used to represent the feature changes of the graph, for example: average distance ($\overline{dist}$), diameter (d), transitivity (t), second small eigenvalue (μ^2) of the Laplace matrix, betweenness centrality (C_B) and closeness centrality (C_C). Under the p-perturbance, the information loss of the social network graph is expressed as V(G,G_p^*). V(G) is the topological property value of the original graph, and V(G_p^*) is the topological property of the graph after the perturbance.

$$V(G, G_p^*) = \left\| V(G) - V(G_p^*) \right\|_2 \tag{3}$$

Table 6 shows the information loss of the PPMPK and RA/D and R-SW algorithms after perturbing the edges under different datasets of p=0.25 and r=3 (the bold value is the optimal value). It can be seen from the table that the PPMPK algorithm can protect the structure information of the graph well in most cases. Figure 8 shows the average shortest path change after anonymity in the jazz social network. The closest algorithm to origin graph is the PPMPK algorithm, which indicates that the information loss of the PPMPK anonymity is the smallest.

Table 6: The information loss situation

Methods	$\overline{\textbf{dist}}$	**d**	**t**	$\mathbf{C_B}$	$\mathbf{C_C}$	μ^2
karate						
R-A/D	0.048	0.331	0.031	0.030	0.053	**0.058**
R-SW	0.120	0.596	0.019	0.022	0.037	0.184
PPMPK	**0.020**	**0.120**	**0.011**	**0.019**	**0.026**	0.060
Jazz						
R-A/D	0.188	1.927	0.111	0.008	0.045	1.856
R-SW	**0.127**	0.504	0.105	0.006	**0.032**	**0.056**
PPMPK	0.237	**0.244**	**0.093**	**0.001**	0.102	0.137
URV email						
R-A/D	0.099	0.508	0.038	0.128	0.147	0.317
R-SW	0.114	0.165	0.044	0.088	**0.012**	**0.003**
PPMPK	**0.073**	**0.100**	**0.042**	0.001	0.039	0.047
Ploblogs						
R-A/D	0.116	2.319	0.040	0.144	0.110	0.429
R-SW	0.088	0.846	0.033	0.089	0.038	0.062
PPMPK	**0.010**	**0.600**	**0.001**	**0.001**	**0.022**	**0.058**

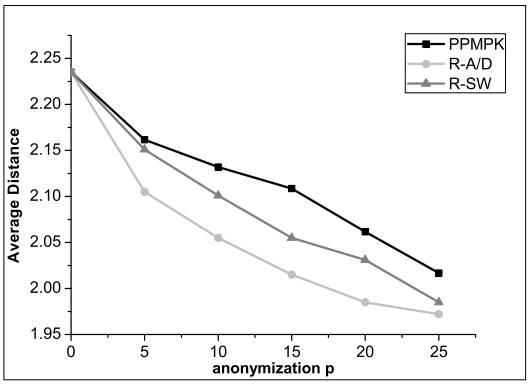


Figure 8: The information loss of average shortest path

6.4 Analysis of Community Structure

By running three kinds of clustering methods to evaluate the influence of the perturbation edge on community, the technology used is python's $\|igraph\|$ package. Community clustering method are Walktrap(WT) [10], Infomap(IM) [2], Multilevel(ML) [3]. Although these algorithms allow overlapping communities, this experiment sets the corresponding parameter to zero. In order to visualize the changes in the community, the precision index (Precision_index) is introduced [11]. If the $l_{tv}(v) = l_{pv}(v)$ is equal to 1. the result indicates that the node in the original graph is the same as the anonymous community. If the node community is not the same as the anonymous community, then $l_{tv}(v) \neq l_{pv}(v)$ is 0. The precision index formula is defined as follows:

$$Precision\_index\,(G, G^*) = \frac{1}{n} \sum\nolimits_{v \in G} \rho_{l_{tv}(v) = l_{pv}(v)} \tag{4}$$

Table 7 shows the community changes of PPMPK, R-A/D and R-SW algorithms after clustering at p=0.25 and r=3. It can be seen from Table 7 that the R-SW algorithm works better than the R-A/D algorithm in protecting the community to which the node belongs, because the R-SW is only switching edge. The PPMPK algorithm considers the node's K-Core for edge perturbation anonymity. The PPMPK algorithm can achieve the best results in most cases. Figure 9 shows the results of the Polblogs dataset using the Infomap clustering algorithm. As the anonymity increases, the accuracy of clustering results decreases. From the overall results, the PPMPK algorithm is more effective than R-SW and R-A/D algorithms in protecting the community.

6.5 Analysis of Large-Scale Data Results

For large datasets, Figures 10 and 11 are the results of the algorithms running in the Amazon dataset. As the anonymity increases, the relative error of graph connectivity and average clustering coefficient increases. This is because the algorithm needs to be perturbed more edges as the anonymity P increases. The R-SW and R-A/D algorithms do not consider the nodes' changes after

Table 7: Perturbed 25% of the edge of community changes

	Method	IM	ML	WT
karate	R-A/D	0.236	0.186	0.327
	R-SW	0.284	0.213	0.323
	PPMPK	**0.114**	**0.117**	**0.147**
Jazz	R-A/D	0.131	0.118	0.113
	R-SW	0.101	**0.099**	**0.078**
	PPMPK	**0.089**	0.142	0.08
URV email	R-A/D	0.313	0.396	0.495
	R-SW	0.251	0.373	0.281
	PPMPK	**0.22**	**0.154**	**0.255**
Polblogs	R-A/D	0.226	0.23	0.227
	R-SW	0.123	**0.062**	0.066
	PPMPK	**0.062**	0.102	**0.023**

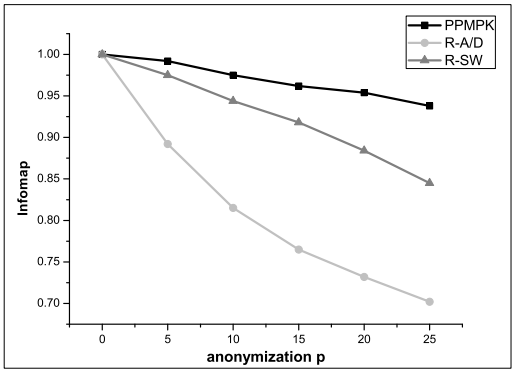


Figure 9: The result of Infomap clustering algorithm

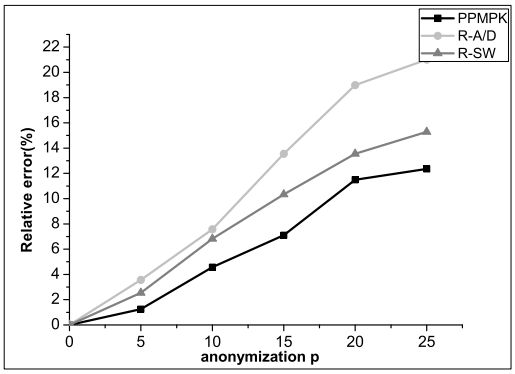


Figure 10: The result of connectivity

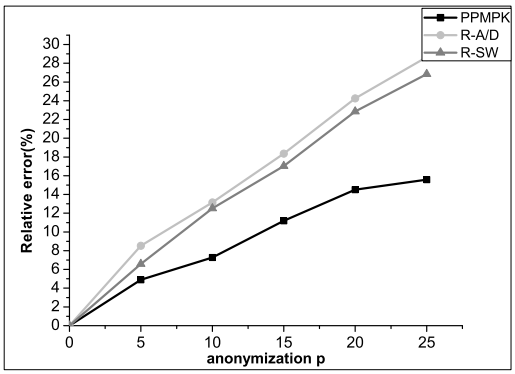


Figure 11: The result of average clustering coefficient

anonymity, so the structural information loss of the graph is larger than the PPMNK algorithm. The information loss of PPMPK algorithm at P=25% is less than 20%, which guarantees the structure of the graph well. It shows that the PPMPK algorithm also has a good performance in large-scale social networks.

7 Conclusion

In order to protect the stability of the node community, a large-scale social network privacy protection method for protecting the k-core is proposed. In order to deal with the large-scale social network graph, the algorithm used Pregel graph calculation model as a basis to compute large-scale social network graphs. The core number was used to measure the influence of the nodes. The core number of nodes were kept constant after anonymity. The community of social network graph were kept stable. Finally, it was verified that the PPMPK algorithm can effectively protect the structural stability of the graph community by experimental tests and analysis while increased the data availability of the graph and ensured the privacy of the social network graph compared with other algorithms.

Acknowledgments

This work is partially supported by National Natural Science Foundation of China (No.61562065) and Inner Mongolia Natural Science Foundation (No.2019MS06001). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

References

[1] R. Assam, M. Hassani, and M. Brysch, "(k, d)-core anonymity: Structural anony-mization of massive networks," in *The 26th International Conference on Scientific and Statistical Database Management*, pp. 1–12, 2014.

[2] V. D. Blondel, J. L. Guillaume, and R. Lambiotte, "Fast unfolding of communities in large networks," *Physics and Society*, vol. 2008, no. 10, pp. 10008, 2008.

[3] B. J. Cai, H. Y. Wang, and H. R. Zheng, "Evaluation repeated random walks in community detection of social networks," in *International Conference on Machine Learning and Cybernetics*, pp. 1849–1854, 2010.

[4] A. Campan, Y. Alufaisan, and T. M. Truta, "Preserving communities in anony-mized social networks," *Transactions on Data Privacy*, vol. 8, no. 1, pp. 55–87, 2015.

[5] A. M. Fard, K. Wang, and P. S. Yu, "Limiting link disclosure in social network anal-ysis through

subgraph-wise perturbation," in *Proceedings of 15th International Conference on Extending Database Technology (ICEDT'12)*, pp. 109–119, 2012.

[6] A. Ford and K. Wang, "Neighborhood randomization for link privacy in social network analysis," *World Wide Web-internet and Web Information Systems*, vol. 18, no. 1, pp. 9–32, 2013.

[7] T. Gao, T. Li, R. Jiang, Y. Ming, and R. Zhu, "Research on cloud service security measurement based on information entropy," *International Journal of Network Security*, nol. 21, no. 6, pp. 1003–1013, 2019.

[8] X. Hui and W. Yang, "Security access solution of cloud services for trusted mobile terminals based on trustzone," *International Journal of Network Security*, vol. 22, no. 2, pp. 201–211, 2020.

[9] S. Kumar and P. Kumar, "Upper approximation based privacy preserving in online social networks," *Expert Systems with Applications*, vol. 88, pp. 276–289, 2017.

[10] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *National Academy of Sciences*, vol. 105, no. 4, pp. 1118–1123, 2008.

[11] F. Rousseau, J. Casas-Roma, and M. Vazirgiannis, "Community preserving anonymization of graphs," *Knowledge and Information Systems*, vol. 54, no. 2, pp. 1–29, 2018.

[12] S. B. Seidman, "Network structure and minimum degree," *Social Networks*, vol. 5, no. 3, pp. 269–287, 1983.

[13] Y. Sun, Y. Yuan, and G. Wang, "Splitting anonymization: A novel privacy-preserving approach of social network," *Knowledge and Information Systems*, vol. 47, no. 3, pp. 595–623, 2016.

[14] X. Ying and X. Wu, "On link privacy in randomizing social network," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 28–39, 2009.

[15] M. Yuan, L. Chen, and P. S. Yu, "Protecting sensitive labels in social network data anonymization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 3, pp. 633–647, 2013.

[16] X. Ying and X. Wu, "Randomizing social networks: A spectrum preserving approach," in *Proceedings of the SIAM International Conference on Data Mining (SIAM'08)*, pp. 739–750, 2008.

[17] X. L. Zhang, J. Li, and J Liu, "Social network sensitive area perturbance method based on firefly algorithm," *IEEE Access*, vol. 7, pp. 137759–137769, 2019.

[18] X. L. Zhang, J. Liu, and J. Li, "Large-scale dynamic social network directed graph k-in&out-degree anonymity algorithm for protecting community structure," *IEEE Access*, vol. 7, pp. 108371–108383, 2019.

[19] L. Zhang and W. Zhang, "Edge anonymity in social network graphs," in *Proceedings of International Conference on Computational Science and Engineering (ICCSE'09)*, pp. 1–8, 2009.

[20] X. L. Zhang, W. C. Zhang, and C. Zhang, "Dgsperturb: A distributed social privacy protection algorithm based on graph structure perturbation," *Journal of Computers (Taiwan)*, vol. 28, no. 5, pp. 51–61, 2017.

[21] X. Zheng, Z. Cai, and G. Luo, "Privacy-preserved community discovery in online social networks," *Future Generation Computer Systems*, vol. 93, pp. 1002–1009, 2019.

Biography

Jian Li received bachelor's degree in software engineering from Inner Mongolia University of Science and Technology, Inner Mongolia, China, in 2017. At present, he is pursuing a master degree in computer science and technology at Inner Mongolia University of Science and Technology, China. His research areas include big data processing technology, social network privacy protection technology.

Xiaolin Zhang received bachelor's degree in computer science and technology from Northeastern University in 1988, and master's degree from Beijing University of Science and Technology in 1995, and Ph.D. in computer science and technology from Northeastern University in 2006. Since 1988, she has worked in Inner Mongolia University of Science and Technology. her research areas include big data processing, social network privacy protection, XML database, XML data stream, wireless sensor network, uncertain database, flame image database.

Jiao Liu received the BS degree in medical information engineering from Taishan Medical College, Shandong, China, in 2017. Currently, she is pursuing a master's degree in computer science and technology at Inner Mongolia University of Science and Technology. Her research interests include big data processing technology, social network privacy protection technology.

Lu Gao received her master's degree from Inner Mongolia Agricultural University. She is an associate professor at Inner Mongolia University of Science and Technology. Her research interests include large-scale social network privacy protection.

Huanxiang Zhang received her master's degree from Inner Mongolia University of Science. She is a Ph.D. student at Shanghai University. Her research interests include large-scale social network privacy protection and artificial intelligence.

Yueyang Feng received the BS degree in mathematics from Tianjin University of Technology and Education, Tianjin, China, in 2017. At present, she is pursuing a master's degree in mathematics at Inner Mongolia University of Technology, China.