Label-bag based Graph Anonymization by Edge Addition

Chongjie LI† Toshiyuki AMAGASA†‡ and Hiroyuki KITAGAWA†‡

† Department of Computer Science, Graduate School of Systems and Information Engineering, University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan

‡ Center of Computer Science, University of Tsukuba 1-1-1 Tennodai, Tsukuba, Ibaraki 305-8573, Japan

E-mail: † lichongjie@kde.cs.tsukuba.ac.jp, ‡ {amagasa, kitagawa}@cs.tsukuba.ac.jp

Abstract The privacy concerns in publishing graph data, such as social-network graphs, has been gaining much public attentions in recent years due to the growing demands for publishing graph data containing privacy. In this paper we address the K-anonymity problem in edge-labeled graphs based on the label-bag model. To this end we provide greedy algorithms, and evaluate them by some experiments.

Keyword Graph Privacy, K-anonymity, Label-bag

1. Introduction

Social networks (SNs) have shown remarkable development in recent years. Due to the rapid proliferation of SNs, there is a growing concern in privacy. SN data publishing is one of the main channels for privacy breaches. SN data owners, such as Facebook and LinkedIn, sometimes have a responsibility to publish their data for various purposes. In this case, transforming data in such a way that privacy information is not released is important. Such data transformation is called "anonymization."

In many cases, SN data is represented as a graph $G = (V, E)$, where vertices (V) represent entities and edges (E) represent relationship among them. To model more complex information, a graph may have labels on vertices and/or edges that describe the attributes of entities or properties of relationship as shown in Figure 1. By having access to this graph, adversaries can gain more information with their existing knowledge about some involved entities (i.e. Bob has an artist friend). To prevent the privacy of entities from being violated, an appropriate graph anonymization method is needed to sanitize the original graph before publishing.

In this paper we address the K-anonymity problem of edge-labeled graphs based on the label-bag model. Since it has been shown that the problem is NP-hard, we provide heuristic methods based on edge-addition in which we try to maintain the utility as much as possible. Additionally, we evaluate the effectiveness of our proposed scheme in some experiments.

The rest of this paper is organized as follows, Section 2 introduces some related works, and we formalize the problem in Section 3. Section 4 gives the proposed method and some discussion of noise vertex is involved in Section 5. We show the results of experiments in Section 6 and make a conclusion in Section 7.

2. Related Work

Privacy-preserving graph publishing is to transform a graph into another in such a way that adversaries with certain background knowledge cannot get the identity and cannot re-identify the entities and/or relations. So far, there have been lots of methods proposed to address the graph anonymization problem.

The problem can be categorized into several classes according to the graph model and quantity of knowledge that adversaries are assumed to have, a main part of researches are focusing on anonymizing structural information. These methods can be roughly divided to two categories: unlabeled graph anonymization [2,3,6,9,10,11,12,15] and labeled graph anonymization [4,7]. Labeled case can be further divided according whether vertices or edges (or both) are labeled. Most early works focus on unlabeled or vertex-labeled cases. Sweeney [2] gives the early idea of K-anonymity by replacing the identifiers of published data. Liu [3] defines the K-degree anonymity that only the degree of certain vertices are known by the adversary. Zhou[6] considers neighborhood attacks of certain vertex and is extended by Tripathy [15] who assumes the adversaries having more information beyond 1-neighborhood knowledge. Later researches such like K-automorphism [9] and K-symmetry [11] propose models with stronger privacy assurance on the whole structure property of graph. Cheng [10] further discusses the k-isomorphism situation with consideration on link information. Recent years, attentions have been paid on edge-labeled problems like Yuan [7] and Kapron

[4].

In terms of the anonymization methods used, there are

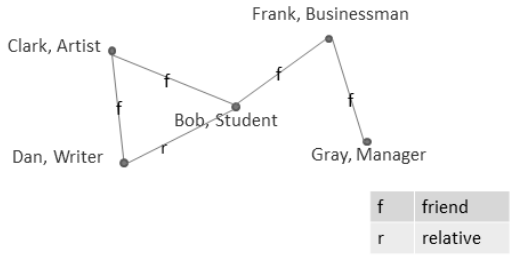

Figure 1: Social Network Graph

proposals focusing on edge addition/deletion [3,4,6,10,15], vertex/edge addition [11,14], vertex generalization [13], edge label generalization [7] and class/cluster-based method [1,12]. Other approaches may be related to active attacks like Backstrom [5], in which case attackers may change the data.

Our work is inspired mainly by [4,7]. However, we focus on edge addition operation instead of label generalization methods in [7], since its core idea is to realize degree k-anonymity and generalize edge labels, which introducing a lot of noises, and it is essentially a direct extension from unlabeled models (noticing that in the early grouping and adding steps, edge labels are not taken into consideration). In some cases we may have restrictions not to modify the edge labels. Kapron [4] gives the proof of finding a result for label sequence based anonymization with minimum cost is NP-hard when $K > 2$. However, only the algorithm for bipartite graph with $K = 2$ is given, which belongs to P. For this reason, in this paper, we provide greedy algorithms based on label-bag graph. Formal definition of the problem is given in the next section.

3. Problem Definition

In this section we formally define the label-bag based graph anonymization problem. First we give the definition of label-bag (LB).

Definition 1 (Label Bag LB): For a vertex v_i, LB_i is the set of all labels on edges which has one end point of v_i.

With the definition of label-bag, we can define the label-bag based $K-anonymity$.

Definition 2 ($LB\ K-anonymity$): For any vertex v of V in graph G, there exists at least $k-1$ vertices with the same label-bag.

For example, Figure 2(b) is an anonymized version of the original graph by replacing identifies (names) with meaningless arbitrary unique numbers. The label-bag of vertex 2 and 3 in Figure 2(b) is *'ab'* and *'abb'*

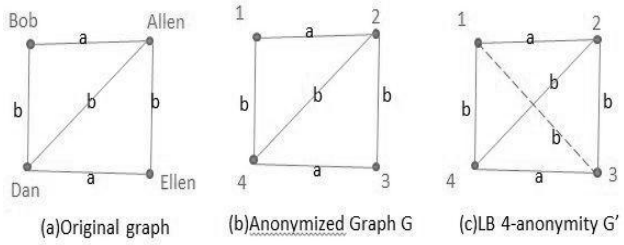

Figure 2: Example of $LB\ K-anonymity$

respectively. Then we are able to give the definition of label-bag based anonymization problem.

Definition 3: Given an edge-labeled graph $G = (V, E)$ and an integer K, find a graph $G' = (V, E\ U\ E')$ that G' satisfies $LB\ k-anonymity$ and makes $|E'|$ minimal.

The above definition is modified from that of [4], where we use the term "label bag", since we do not consider the order among labels, whereas "label sequence" is used in [4].

Figure 2(c) shows how to make a graph $LB\ K-anonymized$ by adding edges to graph G in Figure 2(b). G' satisfies $LB\ K-anonymity$ for any $K \leq 4$. Note that only edge addition is allowed in this setting. In other words, we do not consider other operations, such as edge deletion or perturbation..

4. LB K-anonymization Algorithms

According to the definition of $LB\ K-anonymity$ problem, we attempt to achieve it in the following way: 1) make a division D, where V is dvided into n anonymous groups $\{g_1, g_2, \ldots, g_n\}$ (n is unknown); and 2) make all vertices in same group g_i have the same LB (after edge addition), having $|g_i| >= K$. The objective is to find a division with minimum edges added to the original graph. As already mentioned above, finding optimal solution is an NP-hard problem. So, we propose a two-phase greedy algorithm. Algorithm 1 shows the structure of the basic method.

Algorithm 1: Basic Algorithm Structure

```
1. {S} ← ComputeGroupStrategy(|V|, K)
2. mincost = infinity
3. for each s in S
4.     {g} = GreedyGrouping(s)
5.     cost = EdgeAddition({g})
6.     if cost < mincost
7.         mincost = cost
8.     end if
9. end for
```

First, we decide the group sizes for all groups. (Appendix A introduced the detail in size decision.) Each decision is

named as a strategy. For each strategy we perform a two-phase computation procedure to get the result cost and find the minimum, which will be introduced in Sections 4.a and 4.b, respectively.

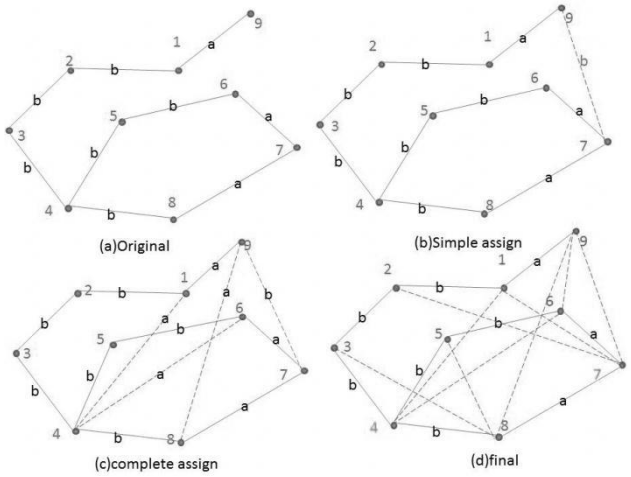

Figure 3: Illustration of workflow

We give some definitions that will be used in the algorithm introduction.

Definition 4 (TLB): For each group g_m, TLB_m is the objective LB that all members in this group are supposed to reach.

Definition 5 (RLB): For every vertex v_i, RLB_i is the difference between TLB_m and LB_i., where v_i belongs to g_m.

Definition 5 can be represented as $RLB_i = TLB_m - LB_i$. Here the operation $"-"$ means to compute the label bag which belongs to mi but not m_j.

a. Greedy grouping

During the 1st phase, we aim to divide vertices into different anonymous groups. We utilize the TLB as a measurement to result in good grouping. Algorithm 2 shows the pseudo-code of greedy grouping.

Algorithm 2: **Greedy Grouping**

Input: *strategy s*
Output: *Grouping* $\{g\}$

1. $\{g\} \leftarrow empty$
2. $for\ m = 1\ to\ s.numOfgroup(m)$
3. $\quad g_m.size = s.grupsize(m)$
4. $\quad |TLB_m| = 0$
5. $\quad for\ i = 1\ to\ gm.size$
6. $\quad\quad find\ v\ in\ V\ with\ least\ |TLB_m|\ increase$
7. $\quad\quad insert\ v\ to\ g_m,\ update\ TLB_m$
8. $\quad\quad i++$
9. $\quad end\ for$
10. $\quad \{G\} \leftarrow \{G\}\ g_m$
11. $end\ for$

Figure 4 gives an example of how the anonymous groups for graph in Figure 3(a) are generated according to algorithm 2. Assume the strategy we get from Algorithm 1 is {3,3,3}, which means we have three groups with size 3. In each step, we choose the vertex with least TLB increase to be added to the group g until it reaches the group size. For example, after we have vertex 9 and 1 in group 1, vertex 6 is the only vertex that would result in no change to the TLB1 and thus being added to this group. Such greedy step is performed until all groups reach the predefined sizes.

v	LB
1	a,b,
2	b,b
3	b,b
4	b,b,b
5	b,b
6	a,b
7	a,a
8	a,b
9	a

Group 1

Member	TLB
9	a
9,1	a,b
9,1,6	a,b

Group 2

Member	TLB
2	b,b
2,3	b,b
2,3,5	b,b

Group 3

Member	TLB
7	a,a
7,8	a,a,b
7,8,4	a,a,b,b,b

Figure 4: Greedy Grouping

Group	Vertex	neighbor	LB	TLB	RLB
1	9	1	a	a,b	b
	1	2,9	a,b		
	6	5,7	a,b		
2	2	1,3	b,b	b,b	
	3	2,4	b,b		
	5	4,6	b,b		
3	4	3,5,8	b,b,b	a,a, b,b,b	a,a
	7	6,8	a,a		b, b, b
	8	4,7	a,b		a,b,b

Figure 5: Grouping Result

As introduced in Appendix A, usually it is impossible to examine all possible strategies and in case of large $|V|$ and K, we randomly pick M strategies to be candidate answers, thus it becomes a tradeoff between execution time and utility cost. Also the chosen of M relates to the ratio of getting answers.

b. Edge addition

After we have assigned vertices to groups, we must add edges to make all groups satisfying the TLB condition. Algorithm 3 is the pseudo-code of edge addition process. Two kinds of assignment operations are performed to achieve final $LB\ K-anonymity$ state. "SimpleAssign" step checks all unconnected pairs of vertices with same RLB label. We can find that both vertices 7 and 9 in Figure 5 have a label *'b'*. So, an edge can be added in between

(Figure 3(b)). However, it is impossible to add an edge to vertices 4 and 8. In this situation, "ComplementAssign"

Algorithm 3: Edge Addition

Input: *Grouping* $\{g\}$
Output: *cost*
1. *Compute TLB and RLB for each group*
2. *do*
3. *SimpleAssign*()
4. *ComplementAssign*()
5. *until: all* $|RLB| = 0$ *or* $|RLB|$ *cannot be decreased*
6. *if* $|RLBm| == 0$
7. $cost \leftarrow |E'|$
8. *end if*

procedure is executed, which is shown in Figure 3(c), where we increase the TLB of group 1 by 'b' and have 3 more edges added.

Since both the two functions are incremental and assure $|RTB_{mi}|$ is decreasing by each step without increasing the $|RTB_{mj}|$ for another group mj (unless the situation of no answer, which would be talked about in Section 5), the algorithm shall terminate in finite time, and maximum loop time is sum of $|RTB|$ for all groups.

c. Clustering-based grouping

Since the edge addition procedure highly depends on the previous stage and the property of a grouping strategy would have great influence to the final cost, a more efficient grouping strategy is needed. Inspired by clustering approaches like Campan [8], we give a clustering based method shown in Algorithm 4 that provides an alternative way of grouping algorithm introduced in Section 4.a.

Algorithm 4: Clustering-based Grouping

Input: *Graph* G, *integer* K
Output: *Grouping* $\{g\}$
1. $\{g\} \leftarrow \{V\}$
2. *do*
3. *merging two closest clusters with condition C*
4. *until: all* $|g| \geq K$ *or no more merging*
5. *GroupAdjust*()

Here condition C refers to:

C: Any clusters with size larger than K are not further merged.

Considering that traditional clustering methods have no restrictions on cluster sizes, this condition can effectively help to reduce the average cluster (group) size and avoid bias clustering result. A prototype-based hierarchical clustering method is used: The group set is initialized with vertex set and we keep merging the closest clusters. The distance between two clusters is defined as:

Dist 1:

$$Dist(g_{mi}, g_{mj}) = \left|TLB_{m_i} - TLB_{m_j}\right| + \left|TLB_{m_j} - TLB_{m_i}\right|$$

The final step would be responsible for the situation when only one cluster is under size K while it cannot be merged to any other one according to $condition\ C$. In that case, we can merge this clusters with a previous cluster which would results in minimum increase to the original value of $|TLB|$.

A major drawback of this clustering grouping method is that group size is not taken into consideration when choosing closest clusters, while in fact it could be much easier to satisfy the same TLB for a group with smaller size than a larger one. So, we give two alternative distance metrics as follows:

Dist 2: $$Dist\left(g_{m_i}, g_{m_j}\right) = Dist1 * \left(\left|g_{m_i}\right| + \left|g_{m_j}\right|\right)$$

Dist 3: $$Dist\left(g_{m_i}, g_{m_j}\right) = (TLS_{m_i} - TLS_{m_j}) * \left|g_{m_j}\right| + (TLS_{m_j} - TLS_{m_i}) * \left|g_{m_i}\right|$$

The comparison would be introduced in Section 6.

d. Complexity analysis

The time complexity for Greedy Grouping method can be easily computed from Algorithm 2, which is $O(N^2)$. By using dynamic lists of arrays to represent the adjacent lists, the memory needed for nodes information is only $O(kN)$, constant k can be set to $2D$ initially, where D is the average degree of vertices. The space needed for group information is also linear to input size, $O(|V|/K)$.

In case of using clustering method, traditional agglomerative hierarchical clustering method requires $O(N^3)$ execution time, which is not applicable to large real data. So. we use the implementation of Fastcluster by Mullner [16], which has most time complexity of $O(N^2)$. We modify the distance metric during procedure according to our setting and we tend not to use the distance matrix to record pairwise distance since it requires $O(N^2)$ space.

In the second phase, the Edge Addition algorithm costs $O(k'N^2)$, coefficient k' can be viewed as iteration time, which is positively correlated to sum of $|RTB|$. This is why grouping phase would have influence on the execution time of Edge Addition.

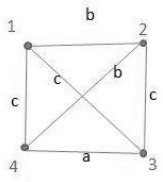

Figure 6

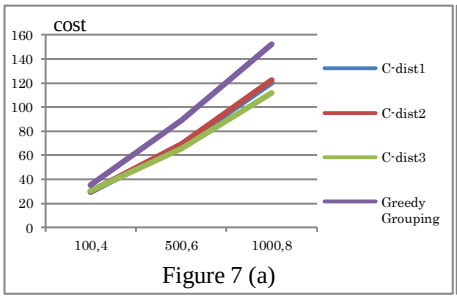

Figure 7 (a)

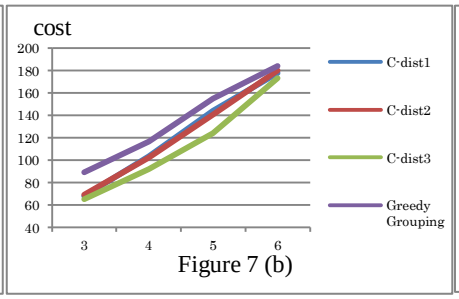

Figure 7 (b)

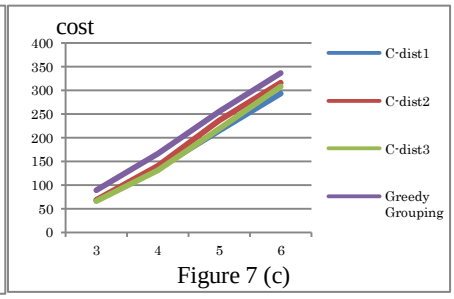

Figure 7 (c)

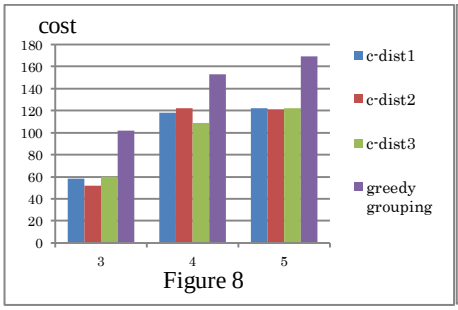

Figure 8

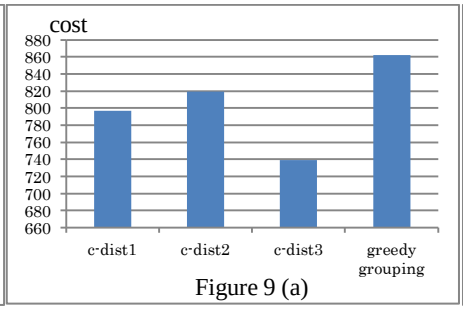

Figure 9 (a)

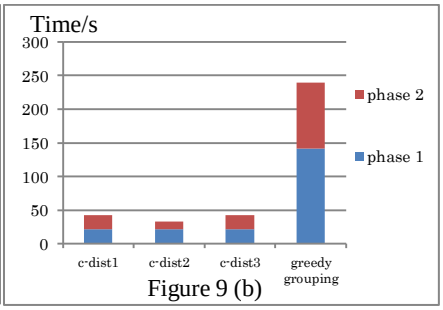

Figure 9 (b)

5. Noise Vertex

It should be noticed that the label-bag based K-anonymity vertex can be added. For example, Figure 6 is a complete graph, which means no more edges can be added. Obviously, this graph does not satisfy $LB\ k-anonymity$ when $k \geq 2$. Since $k = 1$ is meaningless, decreasing K or increasing M in algorithm 2 also fail to solve the problem

To prove an arbitrary graph cannot get a $LB\ k-anonymity$ result is also NP hard, since the worst case requires to try every possible edge-addition operation, which is in O(2^n). One of the ways to issue this problem is to introduce noise vertex.

Her,e we provide a naive way to realize $LB\ k-anonymity$ by edge addition using noise Vertex in Algorithm 5.

Algorithm 5: Edge Addition with Noise Vertex

Input: *Graph G'*
Output: *cost*
1. *Compute TLB and RLB for each group*
2. *foreach label l in {RLB}*
3. *initial a group with ngl in size F*
4. $Q \leftarrow \{v|\ cRLB_i(l) > 0\}$
5. *sort Q in descending order of* $cRLB_i(l)$
6. *while Q is not empty*
7. $u <- Pop(Q)$
8. *add edges between u and vertices in* ng_l
9. *end while*
10 *Complete in* ng_l
11. *end foreach*
12. *return cost*

Let G' be the original graph or Graph state after being problem is not assured to have a result in the case that only edge addition is allowed and no noise

processed Algorithm 2 for purpose of reducing the number of noise vertices added. In line 3, the size F is computed according this formula:

$$F = \begin{cases} 3 & , \quad K = 2,\ \sum_{i \in V} cRLB_i(l) = 1 \\ \max(K,\ \sum_{i \in V} cRLB_i(l)), & else \end{cases}$$

In line 4, $cRLB_i(l)$ is the number of label l in $cRLB_i$. While executing step 8, we always start from vertex with least degree in ng. We give an example in Appendix C.

6. Experiment

We conduct a series of experiments to evaluate the efficiency and utility of our algorithms. Experiments are conducted in environment as Table 1.

CPU	2.26 GHz Intel Core 2 Duo
Memory	4GB 1067MHz DDR3
Language/Compiler	c/gcc-4.2
Data	Synthetic: Small world Graph [17]
	Real_1: Speed Dating Data[18] Real_2: Collaboration network [19]

Table 1: Experimental Environment

We compare the execution time for different grouping methods and choose the number of edges added as the measurement for utility.

Results of synthetic data are shown in Figure 7. Figure 7(a) compares different greedy grouping algorithms and

clustering in varied distance metrics. We use 15 random generated data set having vertex size $|V|$ = 500, average degree D=6, label kinds l=2 and K=3. For greedy grouping algorithm, the iteration time M is set as 5. (Default parameters are the same in following experiments if without specific explanation.) The result shows the average cost of all the datasets. We can observe that clustering based algorithms have less cost than the Greedy Grouping algorithm. Figure 7(b) shows the variance of cost with increase of K. The cost grows with the growth of K because that larger group sizes usually lead to larger values of $|TLB|$. Figure 7(c) shows the different cost while increasing the number of label kinds. The growing trends remain the same and the priority of clustering algorithms is exhibited again.

Figure 8 shows the results of real data 1. The graph is constructed by 551 vertices, 8368 edges and 2 kinds of labels. The result is close to that of synthetic data and again reflects the utility gain by using clustering-based method.

In result of Figure 9 we use a larger dataset extracted from the co-author network on condensed Matter section of arXiv E-Print Archive [19]. The graph consists of 16726 vertices and 47594 edges with the number of labels set as 3. Figure 9(a) shows the number of edges need for each algorithm. Figure 9(b) is the execution time of two phases for each algorithm. The phase 1 time is almost the same (For Greedy Grouping algorithm is the sum of 5 iterations). The phase 2 execution time differs from algorithms due to the grouping result passed by the previous stage. The Efficiency of all algorithms consists with our analysis in Section 4.d.

7. Conclusion

In this paper, we discussed the K-anonymity problem in privacy-preserving data publishing. This is an extension from the unlabeled model and can be applied to many real-world situations. We provide a greedy algorithm based on label-bag model and realize it in two different ways. We evaluate them by some experiments on both synthetic and real data. Through the results, it is proved to be efficient and of good utility. Als,o we investigate how choices in parameters would influent the cost. In consideration of the problem when there doesn't exist an answer, we give an algorithm based on noise vertex.

A lot of other attack models such as K-automorphism [9] and K-symmetry [11] have been proposed in case of unlabeled graph. To extend our model against stronger structural attack is our future work.

Acknowledgement

This research has been supported in part by the Grant-Aid for Scientific Research from JSPS (# 21240005).

Reference

[1] Smriti Bhagat, Graham Cormode, Balachander Krishnamurthy, Divesh Srivastava. Class-based graph anonymization for social network data. VLDB Endowment 2009.

[2] L. Sweeney. k-anonymity: a model for protecting privacy. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems 2002.

[3] Kun Liu, Evimaria Eerzi. Towards identity anonymization on graphs. SIGMOD 2008.

[4] Bruce Kapron, Gautam Srivastava, S. Venkatesh. Social network anonymization via edge addition. ASONAM 2011.

[5] Lars Backstrom, Cynthia Dwork, Jon Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. WWW' 2007.

[6] Bin Zhou, Jian Pei. Preserving Privacy in Social Networks Against Neighborhood Attacks. ICDE 2008.

[7] Mingxuan Yuan, Lei Chen, Philip S.Yu. Personalized privacy protection in social networks. VLDB Endowment 2010.

[8] Alina Campan, Traian Marius Truta. Aclustering approach for data and structural anonymity in social networks. PinKDD 2008.

[9] Lei Zou, Lei Chen,M. Tamer Özsu. k-automorphism: a general framework for privacy preserving network publication. VLDB Endowment 2009.

[10] James Cheng, Ada Wai-chee Fu, Jia Liu. K-isomorphism: privacy preserving network publication against structural attacks. SIGMOD 2010.

[11] Wentao Wu, Yanghua Xiao, Wei Wang, Zhenying He, Zhihui Wang. K-symmetry model for identity anonymization in social networks. EDBT 2010.

[12] Brian Thompson, Danfen Yao. The union-split algorithm and cluster-based anonymization of social networks. ASIACCS 2011.

[13] Michael Hay, Gerome Miklau, David Jensen, Don Towsley, Philipp Weis. Resisting structural re-identification in anonymized social networks. VLDB Endowment 2008.

[14] Sean Chester, Bruce Kapron, Ganesh Ramesh, Gautam Srivastava, Alex Thomo. k-anonymization of social networks by vertex addition. ADBIS2011.

[15] Tripathy, B.K.; Panda, G.K.. A New Approach to Manage Security against Neighborhood Attacks in Social Networks. ASONAM 2011.

[16] Fastcluster. http://math.stanford.edu/~muellner.

[17] Networkx1.6. http://networkx.lanl.gov/index.html.

[18] Speed Dating Data. http://flowingdata.com.

[19] Newman's scientific collaboration network. Mark Newmam. The structure of scientific collaboration networks. PNAS 98, 404-409.

Appendix

A. Compute Group Strategy

Our target is to divide vertex set $|V|$ into several groups within size at least K. We use a backtrack algorithm to numerate all combinations of group sizes that sum to $|V|$. Theoretically, to find the best grouping strategy, it needs computing all possible combinations which is too large for large node set and K. To reduce the total number of combinations, the following theorem can be used which will be proved in Appendix B.

Theorem 1: Every result anonymous groups should be in size of $[K, 2K)$.

By using theorem 1, the group sizes range can be restricted to $[K, 2K)$. As an example, let $|V| = 9$, and K = 3,then two possible strategies are {3,3,3} and {4,5}.

However, it still results in large consumption since the number of strategies grows exponentially and a threshold value M is required to limit the execution time. In our experiments, default M is set to be 5, and we can increase it to achieve better utility when single iteration time is small.

B. Prove of Theorem 1

We prove theorem 1 using the following description: Let original Graph be $G = (E,V)$, $G' = (E \cup E', V)$ is the anonymized graph that satisfy LB K-anonymity, let $\{g\} = \{g_1, g_2, \dots, g_n\}$ be all the anonymized groups. We can assume g_i is one of the groups with size larger than $2K-1$ without loss of generalization. We construct another grouping strategy $\{g'\} = g'_1, g'_2, \dots, g'_n, g'_{n+1}\}$, satisfying (1) $g_j = g'_j | j = 1 \dots n, j \neq i$,(2) $g_i = g'_i \cup g'_{n+1}$,(3)$|g'_i| \geq K, |g'_{n+1}| \geq K$, and the resulted anonymized graph is $G'' = (E \cup E'', V)$. Then if it can be proved that formula $cost(|E'|, g) \geq cost(|E''|, g')$ holds for any combinations of g'_i and g'_{n+1}, theorem 1 holds too. To prove the above inequality, let's first consider E'' equals to E'. Since G' satisfy the $LB\ k-anonymity$, so every vertex in g_i has the same LB, also $for\ \forall u,\ u \in g'_i \cup g'_{n+1} \rightarrow u \in V - \{g'_1, \dots, g'_{i-1}, g'_{i+1,\dots}, g'n\}, \rightarrow u \in V - \{g_1, \dots, g_{i-1}, g_{i+1,\dots}, g'n\} \rightarrow u \in g_i$. We have every vertex in groupg'_i and g'_{n+1} having the same LB, and G'' satisfy $LB\ k-anonymity$ too. So we can always find a graph G'' satisfying $LB\ K-anonymity$ and $cost(|E'|, g)\ cost(|E''|, g')$ (if not, set E'' to be E').

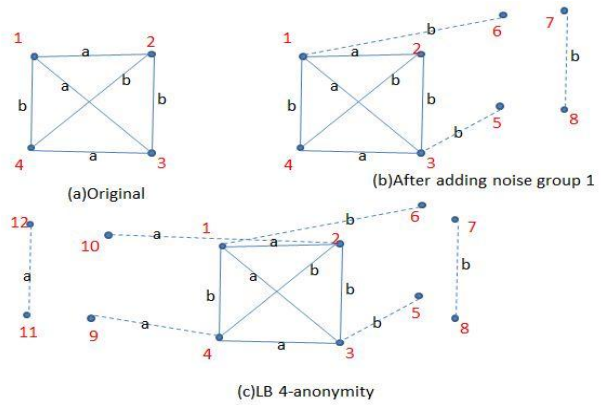

Figure 10 Noise Vertex

C. Example of Edge Addition with Noise Vertex

Figure 10 an example of noise vertex situation. The original graph in Figure 10(a) is a complete graph such that it's impossible to realize $LB\ 4-anonymity$ with only edge addition operation. We illustrates how to use algorithm 5 to achieve $LB\ 4-anonymity$ trough edge addition with noise vertex. First compute the RLB for 10(a) and we have: $RLB_1 = b, RLB_2 = a, RLB_3 = b, RLB_4 = a$. There are two vertices with an label a in RLB, a noise group in size max(2,4) =4 is initialized (vertex 5,6,7,8 in 10(b)), edges between vertices in original graph and noise group (e(1,6),e(3,5)) and make assure the left vertices in same LB with e(7,8). Similarly we deal with label b in same steps and the final $LB\ 4-anonymity$ graph is shown in 10(c). Although to reduce the number of noise vertices added it's better to merge the two groups in this example, the required edges would not decrease. We leave the problem to minimize the noise group size as our future work.