Towards Identity Disclosure Control in Private Hypergraph Publishing

Yidong Li[1] and Hong Shen[1,2]

[1] School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China
[2] School of Computer Science, University of Adelaide, SA, Australia
{ydli,hshen}@bjtu.edu.cn

Abstract. Identity disclosure control (IDC) on complex data has attracted increasing interest in security and database communities. Most existing work focuses on preventing identity disclosure in graphs that describes pairwise relations between data entities. Many data analysis applications need information about multi-relations among entities, which can be well represented with hypergraphs. However, the IDC problem has been little studied in publishing hypergraphs due to the diversity of hypergraph information which may expose to many types of background knowledge attacks. In this paper, we introduce a novel attack model with the properties of hyperedge rank as background knowledge, and formalize the rank-based hypergraph anonymization (RHA) problem. We propose an algorithm running in near-quadratic time on hypergraph size for rank anonymization which we show to be NP-hard, and in the meanwhile, maintaining data utility for community detection. We also show how to construct the hypergraph under the anonymized properties to protect a hypergraph from rank-based attacks. The performances of the methods have been validated by extensive experiments on real-world datasets. Our rank-based attack model and algorithms for rank anonymization and hypergraph construction are, to our best knowledge, the first systematic study for private hypergraph publishing.

Keywords: Identity disclosure control, Private hypergraph publishing, Anonymization, Community detection.

1 Introduction

Identity Disclosure Control (IDC) is a critical problem in private data publishing, and has been widely studied in previous work ([10,9,5,2,11]). Most of these studies focus on preventing entities from background knowledge attacks by modeling a social network as a graph [10,16]. However, the graph-based representation is neither sufficient nor realistic in real-world scenarios.

On the one hand, those potential data buyers (e.g. advertising agencies or application developer) are more interested in attributes reflecting the spending habit of an entity rather than the number of his/her friends. For example, the major purpose for a sport retailer paying for the data from Facebook is to figure

P.-N. Tan et al. (Eds.): PAKDD 2012, Part II, LNAI 7302, pp. 347–358, 2012.

out the entities who are members of a sport interest group. More important, such "interest group" data is usually real for an entity. We fabricate Bob as a member of Facebook, and assume that he is very conscious in protecting private information, such as the date of birth, the living place, the marriage status, and so on. Hence, it is almost unlikely to identify Bob, even the corresponding information is unique. However, as the inherent function and purpose of a social network, Bob will describe his real interests or join certain interest groups without any hesitation.

On the other hand, it is unrealistic to model background knowledge attacks on a social network with the graph-based representation in large-scale networks. For example, we take 1,000 students in Beijing Jiaotong University who have accounts in Renren.com, which is the most popular social network in China. With the graph-based representation, there are only 1.5% students who have unique degrees and 5.5% who have unique neighborhood substructures. However, by considering some properties of the interest groups, the unique rate can climb up to 33%.

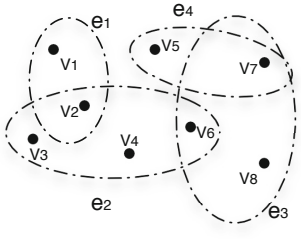

Fig. 1. An example of hypergraphs

In order to remedy the above issues, this paper proposes a hypergraph-based representation (seen in Figure 1) for a social network to depict a set of complex relational entities, such as grouping a population of entities with various attributes. A hypergraph-based representation is a mathematical construction that is quite useful to exploit relationships between different entities [17]. Generally, vertices represent entities and each hyperedge represents a relationship among a set of vertices.

Due to its specific structure, a hypergraph potentially faces more types of breaches with new background knowledge attacks. In our Facebook example, the members and their interest groups are natural to be modeled as a hypergraph and published to third parties without any privacy guarantee. Thus, an attack can be more valid (usually much easier) with the properties of such hypergraphs as background knowledge. For instance, Bob can be easily identified by others knowing his living habits. Notice that, the member-group relation can also be formed as a bipartite graph in our example, but a hypergraph is more general for our discussion, which is suitable for more complex relations such as tripartite

graphs [6]. In this paper, we discuss the IDC problem in social networks with a hypergraph-based representation.

1.1 Our Contributions

- discussing the IDC problem on hypergraphs by modeling rank-based attacks.
- formalizing a general model for *rank-based hypergraph anonymization*, and justifying the hardness of such a perturbation problem.
- proposing an efficient algorithm for rank anonymization, and exploring the issue of constructing a hypergraph with a specified rank set in the first place so far as we know.
- introducing the bias of communities as information loss incurred in hypergraph perturbation.

The remainder of this paper is organized as follows. Section 2 presents a brief survey of IDC on graphs. In Section 3, we model rank attack and introduce efficient metrics for data utility. Section 4 focuses on methods against the proposed rank attack and approaches for hypergraph construction. We present the experimental results in Section 5 and conclude this paper in Section 6.

2 Related Work

The IDC problem has been studied extensively on graphs. As pointed out in [2,9] simply removing the identifiers (or label) of the nodes does not always guarantee privacy. They study a spectrum of adversary external information and its power to re-identify individuals in a social network. The studies in [16,10] extend the above idea by modelling so called neighborhood attack and degree attack respectively. Specifically, the authors in [10] also proposed a two-step framework as property anonymization and graph construction, which is very useful to solve general anonymity problems on graphs.

Zheleva and Getoor [15] considered the problem of protecting sensitive relationships among the individuals in the anonymized social networks. This is closely related to the link-prediction problem that has been widely studied in the link mining community. The work in [14] studies how anonymization algorithms that are based on randomly adding and removing edges change certain graph properties.

Liu *et al.* [11] took weight into consideration for privacy preserving in social networks. They studied situations, such as in a business transaction network, in which weights are attached to network edges that are considered to be confidential. Then, they provide two perturbation strategies for this application. The research in [12] extend the above work by formulating an abstract model based on linear programming. However, the objective of their work still focuses on maintaining certain linear property of a social network by reassigning edge weights.

3 Problem Statement

3.1 Rank Attack

Let V denote a finite set of vertices, and let E be a family of subsets e of V such that $\cup_{e \in E} = V$. Then we call $G(V, E)$ a hypergraph with the vertex set V and hyperedge set E. A hyperedge e is said to be incident with a vertex v when $v \in e$. For a hyperedge $e \in E$, we use *rank* to denote the number of vertices in e, i.e., $r(e) = |e|$. For a vertex v, we use *rank sequence*, denoted as R, to represent the set of ranks of its incident edges $E(v) = \{e_1, e_2, \ldots, e_p\}$, i.e., $R = [r_1, r_2, \ldots, r_p]$. The set of rank sequences for all vertices in V is called the *rank set* of the hypergraph G, which is denoted by $\mathcal{R}_G = \{R_1, R_2, \ldots, R_n\}$, where n is the number of vertices in G.

Table 1. Tables for the example

(a) The adjacency matrix

	e_1	e_2	e_3	e_4
v_1	1	0	0	0
v_2	1	1	0	0
v_3	0	1	0	0
v_4	0	1	0	0
v_5	0	0	0	1
v_6	0	1	0	1
v_7	0	0	1	1
v_8	0	0	0	1

(b) Original $\mathcal{R}$

V	$\mathcal{R}$
v_1	[2]
v_2	[4, 2]
v_3	[4]
v_4	[4]
v_5	[2]
v_6	[4, 3]
v_7	[3, 2]
v_8	[3]

(c) 2-anonymity

V	$\mathcal{R}_2$
v_1	[3]
v_2	[4, 3]
v_3	[4]
v_4	[4]
v_5	[3, 2]
v_6	[4, 3]
v_7	[3, 2]
v_8	[3]

Specially, for a hypergraph being regular, all sequences in the rank set have the same dimension. Without loss of generality, we assume the elements in a rank sequence are sorted as in descending order, $r_1 \geq r_2 \geq \ldots \geq r_p$. Table 1b shows the rank set for the sample hypergraph. Now we model a potential attack on hypergraphs with the properties of edge rank as follows.

Definition 1. *(*rank attack*) Given a hypergraph $G(V, E)$, if the rank sequence R of a vertex $v \in V$ is unique in G, the vertex v can be identified from G by an adversary with the prior knowledge of R, even all vertices and hyperedges unlabelled.*

For example, in Figure 1, we have the corresponding adjacency matrix and the rank set in Table 1a and 1b for the sample. It shows that the vertices v_2, v_6, v_7 and v_8 have unique rank sequences, which have high disclosure risk with a rank attack. Let us take another real case as well: in our Facebook example, Bob is involved in three online interest groups, tennis, cooking and photography, the members in each group are $1,000$, 100 and 10. If the rank set $R = [1000, 100, 10]$ is unique in the hypergraph containing Bob, it is very likely to identify Bob from the published hypergraph.

3.2 Problem Definition

It is clear that the rank set of a hypergraph has to be investigated and modified (if necessary) before published, in order to protect from the above rank attack. We call this problem as *rank-based hypergraph anonymization (RHA)*. The initial idea is to generalize the values of the so-called quasi-identifier in a dataset, which is a group of attributes that can be uniquely identify individuals. In this paper, we define rank-based anonymity with the assumption that all attributes in a rank sequence form the quasi-identifier. However, all of the algorithms proposed later are suitable for the general case as well with a slight modification. Now, we first introduce the term of *rank anonymity* for a hypergraph as follows.

Definition 2. *(k-*rank anonymity*) A hypergraph $G(V, E)$ is k-rank anonymous if for every vertex v, there exist at least $k-1$ other vertices in the hypergraph with the same rank sequence as v.*

For example, from Table 1b, the vertices v_1, v_5, v_3 and v_4 are 2-rank anonymous, while others are all 1-rank anonymous. Therefore, the hypergraph shown in Figure 1 is 1-rank anonymous. By adding v_5 into the hyperedge e_1, the hypergraph G is 2-rank anonymous as shown in Table 1c.

Then, we formally define the RHA problem as follows.

Problem 1. (*rank-based hypergraph anonymization*) Given $\mathcal{R}_G$, the rank set of a hypergraph $G(V, E)$, and an integer k, construct a k-rank anonymous hypergraph $G^{'}(V, E^{'})$, such that the information loss $\mathcal{Z}$ is minimized.

In general, there exist two directions to solve the RHA problem: 1) changing the incident matrix of the hypergraph to adapt the requirement of the rank set, and 2) perturbing the rank set separately and then reconstructing a hypergraph with such modified rank set. The first method has the advantage of maintaining specified data utility globally while ensuring the security. However, such a technique is very inefficient to implement especially for very large hypergraphs. Hence, this paper will follow the second way to perturb a hypergraph, which is described as *rank anonymization (RA)* and *hypergraph construction (HC)* respectively.

3.3 Measuring Quality of Hypergraph Anonymization

Differentiating from some other problems, such as k-anonymity on transactional data, we use a conditional metric $\mathcal{Z} = (\mathcal{Z}; \mathcal{Z}_A)$ to assess the quality of an approach for RHA. The anonymizing cost $\mathcal{Z}_A$ is usually related to the operations of anonymization, while $\mathcal{Z}$ represents one of the important hypergraph property that we suppose to preserve. Here, we can only guarantee a solution of $\mathcal{Z}$ to be optimal for RHA with the condition of certain $\mathcal{Z}_A$. It can be seen as a trade-off between utility and efficiency. In other words, an anonymizing algorithm becomes too complex to implement with a real graph property as $\mathcal{Z}_A$, since it must construct the adjacency matrix to obtain the real property at each step of perturbation.

Anonymizing Cost. As our basic operations for perturbing are to add, delete or reallocate vertices in hyperedges, the method for rank anonymization is naturally required to minimize the changes of hyperedges. Given a hyperedge e and the anonymized $e^{'}$ correspondingly, we define the difference between their ranks as anonymizing cost for a hyperedge, i.e. $|r_e - r_{e'}|$. Then, given a rank set $\mathcal{R}_G$ of a hypergraph $G(V, E)$, we describe the total anonymizing cost as

$$\mathcal{Z}_A = \sum_{i=1}^{m} \sum_{j=1}^{g_i} \|R_{ij} - R_i^*\|^2 \tag{1}$$

where m is the number of anonymized groups, g_i and $R_i^* \in \mathcal{R}_G$ represent the number of objects and the anonymous object in each group.

Information Loss on Community Detection. As a hypergraph is powerful in representing the multi-relationship among vertices, an important and natural requirement is to detect communities in real-world applications [17,13]. Therefore, the methods for RHA also aim at minimizing the effect on community detection on hypergraphs published.

We use a popular metric, called modularity, which is known as a global quality function to identify communities. We revise the definition of modularity in [7] by using the terms of hypergraphs as the cumulative deviation from the random expectation.

$$\mathcal{M} = \sum_{g=1}^{N_C} \left(\frac{\sum_{\{v_i, v_j \in C_g \mid i \neq j\}} c_{ij}}{\sum_s r_s (r_s - 1)} - \frac{\sum_{\{v_i, v_j \in C_g \mid i \neq j\}} d_i d_j}{\left(\sum_s r_s\right)^2} \right), \tag{2}$$

where c_{ij} is the actual number of hyperedges in which i and j are together. Due to space limitations, the induction of Equation 2 is omitted here. Let $\mathcal{M}_G$ and $\mathcal{M}_{G'}$ be the modularity derived from G and $G^{'}$ respectively. Then, we can define the *modularity bias* as information loss,

$$\mathcal{Z}_M = \frac{|\mathcal{M}_G - \mathcal{M}_{G'}|}{\mathcal{M}_G}. \tag{3}$$

4 Algorithms

In this section, we propose algorithms for the RHA problem. It first states the hardness of RHA with anonymizing cost as the objective function, and introduces an efficient heuristic method based on the information loss defined in Equation 1. Then, we discuss hypergraph construction with a specified rank set, which is rarely mentioned in previous studies so far as we know.

4.1 Rank Anonymization

From the definition of RHA, it is obvious that rank anonymization, as the first step, is an optimization problem. The following theorem shows that such an optimization problem is NP-hard, even for the simplest case that all rank sets are with size 2.

Algorithm 1. The Rank Anonymization Algorithm

Input: A set of rank sets $\mathcal{R}$ and an integer k.
Output: An anonymized set $\mathcal{R}^{'}$.
1: **Initialization.**
1.1 find v_s and v_t in V with the most distance in $\mathcal{R}$;
1.3 form groups g_s and g_t containing v_s and v_t with their $k-1$ closest vertices respectively;
1.4 determine anonymous objects o_s and o_t and compute information loss for each group.
2: **Recursion**.
2.1 set all remaining vertices as 1-element group and initial the anonymous object as itself;
2.2 merge two groups with the lowest information loss;
2.3 re-calculate anonymous objects for each group;
2.4 go to 2.2 until every vertex is assigned to a group with size $[k, 2k)$.
3: **Perturbation**.
3.1 replace elements in each group by anonymous object;
3.2 merge all groups as $\mathcal{R}^{'}$ and return.

Theorem 1. *The optimal rank anonymization problem is NP-hard.*

Limited by space, we omit the formal proof (seen in the extended version of this paper). To guarantee the complexity in polynomial time, we introduce an efficient heuristic algorithm as a solution in Algorithm 1. This algorithm is similar with a family of data-oriented heuristics for microaggregation proposed in [3], while the major difference is the objective function due to anonymity.

The computational complexity of Algorithm 1 is $O(n^2 log\frac{n}{k})$. Here, we form a symmetric $n \times n$ distance matrix that each entry represents the Euclidean distance between two rank sets in $\mathcal{R}$. It reduces the complexity of the initialization step to linear. In each step of recursion, the algorithm introduce $O(n^2)$ operations to calculate all new distances among groups. Finally, there are $log\frac{n}{k}$ recursions due to the group merging. Therefore, the total complexity is $O(n^2 log\frac{n}{k})$.

4.2 Hypergraph Construction

Our next task for RHA is to reconstruct a hypergraph with a perturbed rank set $\mathcal{R}$. Some existing work [8,10] has studied popular construction techniques according to various properties of a graph, such as degree and spectrum. Unfortunately, most of these studies only consider graphs, and it has more concerns to apply the proposed methods on hypergraphs due to the computational complexity. The major reason is that one modification on an edge will affect a group of vertices rather than only two vertices in graphs. Specifically, there are two major challenges for the RHA problem: 1) the anonymized rank set has high possibility that is not realizable; and 2)the constructed hypergraph need to maintain the original community.

To explain the first challenge, we define the realizability of a rank set as follows.

Definition 3. *(*Realizability*) A rank set $\mathcal{R}$ is called realizable if and only if there exists at least one hypergraph $G(V, E)$ that has the exact same rank set with $\mathcal{R}$.*

This definition is extended from the realizability of degree on graph construction [4]. We state the following necessary and sufficient condition for a rank set to be realizable.

Lemma 1. *A rank set $\mathcal{R}$ is realizable if and only if, for any entry r_{ik} it holds αr_{ik} $(\alpha = 1, 2, \ldots)$ different vertices in $\mathcal{R}$ containing the same entry with r_{ik}.*

Proof. The sufficiency is obvious. As a hyperedge with rank r contains r vertices, there at least exists $\alpha = 1$. For the necessity, assuming a rank set has αr different vertices having an element with value r, it is easy to form α hyperedges with rank as r.

For example, in Table 1c, for $r = 2, 4, 3$, it holds $\alpha = 1, 1, 2$ respectively, and the 2-anonymized rank set is realizable. However, if we modify R_1 and R_8 to $[3, 2]$ and $[4]$, the rank set is still 2-anonymized but unrealizable with $\alpha = \frac{3}{2}, \frac{5}{4}, \frac{5}{3}$.

Apparently, an anonymized rank set $\mathcal{R}$ has very high probability that it is not realizable. Thus, Lemma 1 introduces a principle in how to develop a construction method for RHA to ensure the success of construction. The basic idea behind is to generate a realizable rank set from $\mathcal{R}$ based on Lemma 1 with minimal modification. Algorithm 2 takes a specified rank set $\mathcal{R}$ as inputs and returns a successfully constructed graph and an *approximate error* σ, which denotes the modification bias of the rank set. Steps from 3 to 9 describe a procedure to remove edges by matching each element in its rank set. Step 5 is a basic search to find all vertices containing the same rank with r_{ri}. Step 6 is crucial to modify a rank set to be realizable based on Lemma 1. Step 11 is to ensure the connectivity of the output hypergraph. If the algorithm terminates and outputs a hypergraph, then this hypergraph has the approximate specified rank set $\mathcal{R}$.

The computational complexity of Algorithm 2 is $O(n^2m^2)$, where n is the number of vertices and m is the maximal degree for all vertices. For each vertex v_i, there are maximal m hyperedges connecting v_i with other nodes. And for each hyperedge, the worst case is traversing all remaining vertices to find $S(v_r)$, which is $n \times m$ times. As there are n vertices, the total complexity is $O(n^2m^2)$.

In Algorithm 2, the basic operations are adding/deleting vertices in each hyperedge of the original graph to satisfy the privacy requirement. However, such an operation may affect the progress of finding communities. Thus, we provide a *community preserving procedure* aiming at minimizing the change of the community set $\mathcal{C}_G$. Our main idea is to first assign a two-way label for each vertex $v \in V$ in $G(V, E)$ according to the community and the min-cut that contain it. Then, we perform vertex addition or deletion only in its incident domain(s). For example, assuming that a hypergraph G has two non-overlapping communities C_1 and C_2 with the min-cut S, a label (C_1, S_v) for a vertex $v \in V$ implies $v \in C_1$

Algorithm 2. The Hypergraph Construction Algorithm

Input: A hypergraph $G(V, E)$ and an anonymized rank set $\mathcal{R}$.
Output: A hypergraph $G^{'}(V, E^{'})$ and the approximate error σ.
1: $V \leftarrow \{v_1, \ldots, v_n\}$, $E \leftarrow \emptyset$, $count \leftarrow 0$;
2: **while** $\mathcal{R}$ consists of non-zero elements **do**
3: pick a random vertex v_r with $R_r \neq \mathbf{0}$;
4: **for** $i \leftarrow 1$ to d_r **do**
5: find a set $S(v_r) := \{v_s \in V \mid \exists r_{sj} = r_{ri}\}$;
6: modify r_{ri} and all r_{sj} as $s \leftarrow |S(v_r)|$ in $S(v_r)$;
7: $\sigma \leftarrow \frac{|r_{ri}-s|}{r_{ri}}$;
8: form an edge e containing all vertices in $S(v_r)$;
9: $E \leftarrow E \cup e$, $r_{ri}, r_{sj} \leftarrow 0$;
10: $V \leftarrow V \cup v_r$;
11: amend the connectivity of $G^{'}$;
12: return $G^{'}(V, E^{'})$ and σ.

and $v \in S_v$. We also use S_0 as a virtual set to denote a vertex does not appear in any min-cuts. Therefore, in Algorithm 2, we can perform the selection of v_s within the domain where the elements have the same label. Apparently, this procedure is application-oriented since there exist a number of algorithms for community detection. However, this limitation can be released in the real-world applications, which the data publisher can make consistent standards on the methods of community detection with data users.

5 Experiments

The experiments are conducted on a 2.16 GHz Intel Core 2 Duo Mac with 4GB of 667MHz DDR2 SDRAM running the Macintosh OS X 10.5.8 operating system. All algorithms are implemented using Matlab 7.0.

We use three real-world datasets, named Mushroom, Nursery and Msweb, which contains $8,124$, $12,960$ and $32,711$ vertices respectively, and 22, 8 and 294 attributes respectively. Specifically, each attribute takes only a small number of values, each corresponding to a specific category. In our experiments, we constructed a hypergraph for each dataset, where attribute values were regarded as hyperedges. Therefore, the Mushroom data includes 122 hyperedges, while Nursery and Msweb have 27 and 294 hyperedges respectively. All three datasets are from the UCI Machine Learning Repository [1].

5.1 Rank Attack on Real-World Data

Our first experiment is to show whether rank attack may happen on real-world datasets. We detect the possibility of rank attack on the test data with varying a specified parameter β, which is a threshold to assess a breach. That is, while the number of vertices sharing the same rank sequence is no larger than β, these vertices are recognized to be disclosed.

Table 2. Rank Attack on Real-World Data

	Disclosure rate (%)			
	$\beta = 1$	$\beta = 3$	$\beta = 5$	$\beta = 10$
Mushroom	16.18	20.24	38.36	55.14
Nursery	4.26	6.21	8.15	13.11
Msweb	4.55	7.59	10.61	13.64

Table 2 reports the percentage of vertices which can be successfully identified by rank attack. It clearly shows that the rank attack indeed be a real issue for hypergraph publishing. All testing datasets have relatively high risk of entity disclosure. For Mushroom data, the disclosure rate of rank attack with $\beta = 10$ is even high as 55.14%, which implies over half of its vertices can be uniquely identified. The rate is 4.55% with $\beta = 1$ for The Msweb contains around 600 individuals have high disclosure risk in the data with the rate 4.55% corresponding to $\beta = 1$. Also, the disclosure rate grows very quick as β increases. For example, the rate on the Nursery dataset increases nearly 10% with $\beta = 10$ than that with $\beta = 1$.

5.2 Impact on Anonymizing Cost $\mathcal{Z}_A$

In this section, we assess the cost incurred in applying various strategies for rank anonymization. As a comparison, we also implement a greedy anonymizing algorithm for rank attack, called GreedyRA.

The graphs in Figure 2 describe the relations between anonymizing cost $\mathcal{Z}_A$ and various k for the Mushroom, Nursery and Msweb datasets respectively. The results show that $\mathcal{Z}_A$ increases slowly while k is not large (e.g. $k < 20$) for both anonymizing algorithms. Furthermore, the GreedyRA arises much higher cost than the RA algorithm does in all cases as expectation. The biggest difference occurs in Msweb, which is over two times for every plot. In addition, the outcomes reveal the efficiency of GreedyRA with small values of k. For example, the costs by the two methods are very close to each other when $k < 50$ in the Nursery data. Usually, the indistinguish level is not required to be very high in the real-world applications. Thus, both anonymizing algorithms work efficiently in such cases.

5.3 Impact on Information Loss

The final experiment is to explore the relation between the modularity bias defined in Equation 2 and k. Figure 3 shows the relative changes of $\mathcal{Z}_M$with HCCP-RA, HC-RA and HC-RAG approaches by varying k. The modularity bias rises up as k increases that follows the similar trend of $\mathcal{Z}_H$. However, the gradients are not steep as that of $\mathcal{Z}_H$ especially when k is not large. This implies

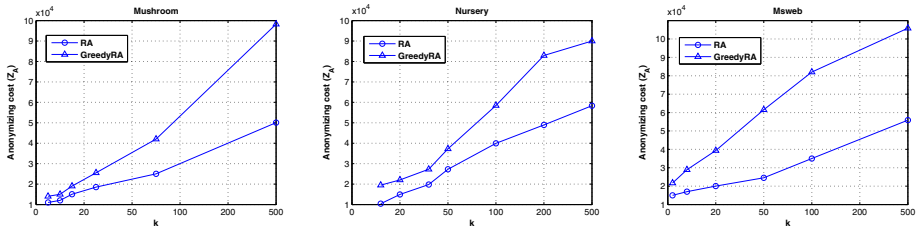

Fig. 2. The relation between $\mathcal{Z}_A$ and k

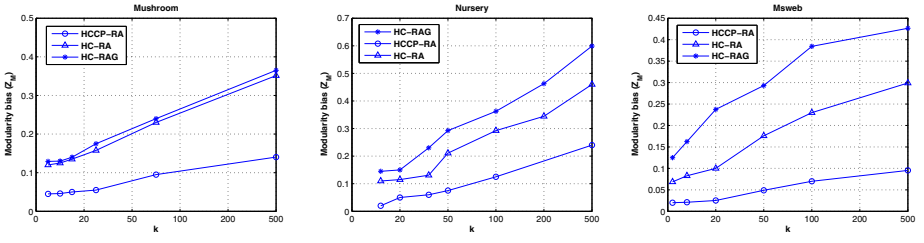

Fig. 3. The relation between $\mathcal{Z}_c$ and k

that the impact of perturbation on modularity is not significant as the intra-group error, as $\mathcal{Z}_M$ is a global measurement. Also, HCCP-RA shows much better performance than the others in all cases as expected. Moreover, the Mushroom data has an interesting result that HC-RA and HC-RAG produce very close plots with each other compared to the result of $\mathcal{Z}_H$. The reason is still not clear and we suppose it is related to certain structure properties of the dataset itself.

6 Conclusion

In this paper, we explored identity disclosure control in private hypergraph publishing. We addressed the problem of rank-based hypergraph anonymization by modeling a novel background knowledge attack with rank. We proposed an efficient heuristic algorithm for rank anonymization, which is shown NP-hard. We also studied the problem of constructing a hypergraph with a specified rank set, and provided methods maintaining the utility of community detection from the original hypergraph.

There are many issues of this work that need to be addressed in further research. As an NP-hard problem, it is worth to develop approximation algorithms for RHA. Also, it is worth to investigate how the proposed approaches affect other real hypergraph properties, such as diameter.

References

1. Asuncion, A., Newman, D.J.: UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences (2010)
2. Backstrom, L., Dwork, C., Kleinberg, J.: Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In: WWW 2007: Proceedings of the 16th International Conference on World Wide Web, pp. 181–190. ACM, New York (2007)
3. Domingo-ferrer, J.: Efficient multivariate data-oriented microaggregation. The VLDB Journal 15, 355–369 (2006)
4. Erdos, P., Gallai, T.: Graphs with prescribed degrees of vertices. Mat. Lapok 11, 264–274 (1960)
5. Feder, T., Nabar, S.U., Terzi, E.: Anonymizing graphs (2008)
6. Ghoshal, G., Zlatiić, V., Caldarelli, G., Newman, M.E.J.: Random hypergraphs and their applications. Phys. Rev. E 79(6), 066118 (2009)
7. Guimera, R., Sales-Pardo, M., Nunes Amaral, L.A.: Module identification in bipartite and directed networks. Physical Review E 76(036102) (2007)
8. Halbeisen, L., Hungerbuhler, N.: Reconstruction of weighted graphs by their spectrum. Eur. J. Comb. 21(5), 641–650 (2000)
9. Hay, M., Miklau, G., Jensen, D.: Anonymizing social networks. Technical Report 07-19, University of Massachusetts Amherst (March 2007)
10. Liu, K., Terzi, E.: Towards identity anonymization on graphs. In: SIGMOD 2008: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 93–106. ACM, New York (2008)
11. Liu, L., Wang, J., Liu, J., Zhang, J.: Privacy preservation in social networks with sensitive edge weights. In: 2009 SIAM International Conference on Data Mining (SDM 2009), Sparks, Nevada, pp. 954–965 (April 2009)
12. Egecioglu, O., Das, S., El Abbadi, A.: Anonymizing weighted social network graphs. In: The 26th International Conference on Data Engineering, ICDE 2010 (2010)
13. Vazquez, A.: Finding hypergraph communities: a bayesian approach and variational solution. Journal of Statistical Mechanics: Theory and Experiment (July 2009)
14. Ying, X., Wu, X.: Randomizing social networks: a spectrum preserving approach. In: SDM 2008: The SIAM International Conference on Data Mining, Atlanta, GA (April 2008)
15. Zheleva, E., Getoor, L.: Preserving the Privacy of Sensitive Relationships in Graph Data. In: Bonchi, F., Malin, B., Saygın, Y. (eds.) PInKDD 2007. LNCS, vol. 4890, pp. 153–171. Springer, Heidelberg (2008)
16. Zhou, B., Pei, J.: Preserving privacy in social networks against neighborhood attacks. In: ICDE 2008: The 24th International Conference on Data Engineering, pp. 506–515. IEEE Computer Society, Los Alamitos (2008)
17. Zhou, D., Huang, J., Scholkopf, B.: Learning with hypergraphs: Clustering, classification, and embedding. Advances in Neural Information Processing Systems 19, 1601–1608 (2007)