Anonymizing Weighted Social Network Graphs

Sudipto Das, Ömer Eğecioğlu, Amr El Abbadi

Department of Computer Science, University of California, Santa Barbara
Santa Barbara, CA 93106-5110, USA
{sudipto, omer, amr}@cs.ucsb.edu

***Abstract*— The increasing popularity of social networks has initiated a fertile research area in information extraction and data mining. Although such analysis can facilitate better understanding of sociological, behavioral, and other interesting phenomena, there is growing concern about personal privacy being breached, thereby requiring effective anonymization techniques. In this paper, we consider edge weight anonymization in social graphs. Our approach builds a linear programming (LP) model which preserves properties of the graph that are expressible as linear functions of the edge weights. Such properties form the foundations of many important graph-theoretic algorithms such as *shortest paths, k-nearest neighbors, minimum spanning tree,* etc. Off-the-shelf LP solvers can then be used to find solutions to the resulting model where the computed solution constitutes the weights in the anonymized graph. As a proof of concept, we choose the *shortest paths problem*, and experimentally evaluate the proposed techniques using real social network data sets.**

I. Introduction

Social Networks have become increasingly popular applications in Web 2.0. Social networking sites such as MySpace, and Facebook have millions of registered users, and each user is associated with a number of others through friendship, professional association (being members of communities), and so on. The resulting graph structures have millions of vertices (users or social actors) and edges (social associations). Recent research has explored these social networks for understanding their structure [1], [2], [3], advertising and marketing [4], and others [5]. As a result, companies (such as Facebook) hosting the data are interested in publishing portions of the graphs so that independent entities can mine the data. In order to protect the privacy of the users against different types of attacks [6], [7], graphs should be anonymized before they are published. Consequently, there has also been considerable interest in the anonymization of graph structured data [8], [9], [10]. But most of the existing research on anonymization techniques tend to focus on *unweighted* graphs for *node* and *structural anonymization*, with very little work concentrating on *edge weight anonymization* [11].

Recently, there has been considerable interest in the analysis of the *weighted* network model where the social networks are viewed as weighted graphs. The weighted graph model is used for analyzing the *formation of communities* within the network [12], *viral and targeted marketing and advertising* [4], *modeling the structure and dynamics* such as opinion formation [13], and for analysis of the network for *maximizing the spread of information* through the social links [14], in addition to the traditional applications on weighted graphs such as *shortest paths*, *spanning trees*, *k-Nearest Neighbors (kNN)* etc. The semantics of the edge weights depend on the application (such as users in a social network assigning weights based on "degree of friendship", "trustworthiness", "behavior", etc.), or the property being modeled (such as detection of communities [12] or modeling network dynamics [13]).

Edge-weight anonymization: why do we care? *First*, even though in most cases node identities are anonymized, there are a number of instances where they are public knowledge. For example, in academic social networks [15], [9], node identities and link structure are public knowledge, but edge weights are sensitive. *Second*, even in the case where the node identities are anonymized, edge weight anonymization is still important since if an adversary re-identifies a node in the anonymized graph, even more information will be revealed if edge weights are not anonymized.

Privacy preserving modeling. Our solution to the problem of edge weight anonymization is to model the weighted graph based on the property to be preserved, and then reassign edge weights to obtain the anonymized graph satisfying the model. To be specific, we preserve *linear* properties of the graph:

Definition 1: A **linear property** of a graph is a property expressible in terms of inequalities involving linear combinations of edge weights.

If we consider that the anonymized graph preserves the structure of the original graph, the objective of the privacy preserving model can be formally stated as:

Objective 1: To construct a model that **correctly** captures the inequalities that must be obeyed by the edge weights for the **linear property** being modeled to be preserved. Any solution to such a model would ensure anonymization of edge weights, while preserving the linear property under consideration.

II. Abstract Model

Abstract model formulation. Our proposed model is based on the observation that a gamut of interesting properties are expressible in terms of linear combinations of edge weights. We now introduce in abstract the technique used for modeling *linear properties* and use Kruskal's algorithm for *minimum spanning tree (MST)* [16] as an example of the algorithm being modeled. The goal of the model is to capture the dynamic behavior of the algorithm using a system of linear inequalities. Given the original weighted graph $G = (V, E, W)$ with positive edge weights represented by variables $x_1, x_2, \ldots, x_m$ (where each x_i corresponds to an edge $i = (u, v) \in E$), our goal is to model the system of linear inequalities in terms of these variables. For example at every step of Kruskal's algorithm [16] for the MST, the edge with the minimum weight amongst the set of remaining edges is selected, and if this edge does not result in a cycle, it is added to the MST. Let (u,v) be the edge selected in the i^{th} iteration, and (u',v') be the edge selected in the $(i+1)^{th}$ iteration, then this implies that $w[u,v] \leq w[u',v']$. If $x_{(u,v)}$ and $x_{(u',v')}$ are the variables representing these edges in the model, then this outcome is modeled by the inequality $x_{(u,v)} \leq x_{(u',v')}$. Therefore, for every pair of edges (u,v) and (u',v') selected in consecutive iterations, the inequality $x_{(u,v)} \leq x_{(u',v')}$ can be added to the model whenever the given weights satisfy $w[u,v] \leq w[u',v']$.

The algorithm makes decisions based on the actual numerical values of the edge weights (or $w[u,v]$'s) and we model this decision in terms of the variables $x_{(u,v)}$. Decisions made at each step of the algorithm can similarly be expressed as inequalities involving the edge-weights. Thus, the execution of the algorithm processing the graph can be modeled as a set of linear inequalities involving the edge weights as *variables*, and this results in a system of linear inequalities:

$$\underbrace{\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1} & a_{k2} & \cdots & a_{km} \end{pmatrix}}_{\mathbb{A}} \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix}}_{\mathbb{X}} \leq \underbrace{\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}}_{\mathbb{B}} \tag{1}$$

If the edge weights are reassigned as any solution of the system of inequalities in (1), this would ensure that the properties of the graph remain unchanged w.r.t the algorithm being modeled. The model can therefore be formulated as a Linear Programming (LP) problem

$$\begin{array}{rl} \text{Minimize (or Maximize)} & F(x_1, x_2, \ldots, x_m) \\ \text{subject to} & \mathbb{A}\mathbb{X} \leq \mathbb{B} \end{array}$$

where F is a linear objective function. Any property that can be expressed as a function of a linear combination of edge weights can be expressed as a Linear Optimization problem, and hence this abstract modeling technique can be used for any such property. Once the model has been developed, any off-the-shelf LP solver package can be used to find a solution to the set of inequalities (constraints) that optimizes F. The model is said to be **correct** if the property being modeled is preserved across anonymization, i.e., any solution to the model ensures that the property being modeled is the same in the original graph as well as the anonymized graph. The **complexity** of the model is the number of inequalities necessary to define the model. Columns in matrix $\mathbb{A}$ correspond to variables in the system, i.e., the number of edges in the graph, and rows correspond to the inequalities produced by the model. The fewer the constraints required by the model, the more efficient it is. Note that most social network graphs are sparse, and hence matrix $\mathbb{A}$ is also sparse, and LP solvers optimized for such large systems can be used. We remark that our technique is not dependent on the semantics of edge-weights, and is general enough to encompass any algorithm based on *linear properties* of the graph.

III. SINGLE SOURCE SHORTEST PATHS

In this section, we demonstrate how the abstract model described in Section II can be used for *single source shortest paths tree*. Given a weighted graph $G = (V,E,W)$, and a source vertex v_0, a *single source shortest paths tree* is a spanning tree of the graph where the path from the source to any other vertex in the tree is the shortest path between the pair in G. Dijkstra's algorithm [17] is a well known greedy algorithm for *single source shortest paths tree*. Given a start vertex v_0, at every step the algorithm selects the vertex u with the smallest known cost from v_0. The algorithm tries to "relax" the neighbors of u by checking if the cost from the source has now decreased because of the selection of u. Due to space limitations, the pseudocode and proofs of the theorems have been moved to an extended version of the paper [18]. In the following discussion, the cost of the path from the vertex u to v is denoted as $D[u,v]$, and $f(u,v)$ is $\sum_{(u',v') \in P[u,v]} x_{(u',v')}$, where $P[u,v]$ denotes the path from u to v in G.

A. Linear model

Dijkstra's algorithm [17] makes a number of decisions based on the outcome of comparisons of linear combinations of edge weights. These decisions can be modeled using the following three categories of inequalities:

- **Category I**: When processing edge (u,v), if $D[v_0,v]$ can be improved, then $D[v_0,v] > D[v_0,u] + w[u,v]$, add constraint $f(v_0,v) > f(v_0,u) + x_{(u,v)}$.
- **Category II**: When processing edge (u,v), if $D[v_0,v]$ can not be improved, then $D[v_0,v] \leq D[v_0,u] + w[u,v]$, add constraint $f(v_0,v) \leq f(v_0,u) + x_{(u,v)}$.
- **Category III**: When extracting the minimum weight edge u for the next iteration, if u' is the previous vertex processed, then $D[v_0,u'] \leq D[v_0,u]$, add constraint $f(v_0,u') \leq f(v_0,u)$. This captures the order in which the vertices are selected.

Theorem 1: A model built from all the inequalities of Categories I, II, and III combined will correctly model Dijkstra's algorithm, i.e., any solution to the model used to anonymize edge weights in the graph results in the same shortest paths tree in the original as well as the anonymized graph.

Complexity of the Model. Category I and Category II combined will result in $O(dn)$ inequalities. This is because, when an edge is processed, either the path to its neighbor is improved (Category I), or it remains unchanged (Category II), and hence every edge results in at least one inequality. Since the average degree per node is d, the resulting number of inequalities is $O(dn)$. The number of inequalities for Category III is $O(n)$ since one inequality of Category III is generated for every vertex processed. Thus, the complexity of the model is $O(dn)$. Since most large real graphs are sparse, i.e., $d \ll n$ (generally d is of the order of tens or hundreds), we refer to this model as the *Linear model* with complexity growing linearly with n.

B. Reduced model

We now improve the performance of the model explained by reducing its complexity. Note that even though Dijkstra's algorithm tries to relax the neighbors when processing a vertex, the ultimate goal is to select an appropriate vertex for the next iteration, i.e., the vertex with the smallest known cost from the source. It does not matter how many times the cost of the path to a particular vertex is improved, the minimum amongst these costs determines its order of selection, and hence the shortest path from the source. Category III inequalities model this information in an efficient way, and hence ideally, only Category III are needed. However Category III inequalities only include the edges that are part of the shortest paths tree. Therefore, if **only** Category III inequalities are considered in the model, then only part of the total number of edges are modeled. Such a model does not put constraints on non-tree edges, and thus, if no care is taken while reassigning edge weights in the anonymized graph, it can lead to violations of the order in the anonymized graph. For instance, if edge (u, v) is a non-tree edge, then a model using only Category III would not impose any constraint on (u, v). Hence a reassignment of weights in the anonymized graph might assign the edge (u, v) a weight such that Dijkstra's algorithm executing on the anonymized graph selects (u, v) as a tree edge. Therefore, to ensure correctness, the model must be augmented to make sure that the non-tree edges are not included in the tree when the algorithm executes on the anonymized graph. The following theorem formalizes this proposition.

Theorem 2: A model which ensures that (i) the order of selection of vertices remains the same even after anonymization, and (ii) non-tree edges in the original graph are not included in the tree constructed on the anonymized graph, will also ensure that the shortest paths tree in the original and anonymized graph are also same, i.e., the model is correct.

Augmenting the model – Complexity and Correctness. Category III inequalities enforce condition (i) of Theorem 2. A simple solution to ensure that condition (ii) is also satisfied is to add all the non-tree edges into the constraints of the model. This can be done as follows: let v_l be the last vertex to be processed by Dijkstra's algorithm, and let T_s represent the shortest paths tree obtained as output from the algorithm, then add all inequalities of the form:

$$\forall(u, v) \in E \wedge (u, v) \notin T_s,$$
$$AddConstraint(x_{(u,v)} > f(v_s, v_l)) \tag{2}$$

This ensures that any path which includes these non-tree edges will have a cost greater than the corresponding path involving only the edges in T_s, and hence all such paths with non-tree edges will not be selected by Dijkstra's algorithm running on the anonymized graph. The $O(n)$ edges in T_s are modeled by Category III inequalities, and the remaining $O(dn)$ edges are modeled by the inequalities in (2). Thus, the complexity of the model still remains $O(dn)$, even after eliminating inequalities of Categories I and II. Note that the inequalities in (2) add very little to the model except for ensuring that any non-tree edge should be assigned a weight that is greater than $D'[v_s, v_l]$, and it does not really matter what weight is assigned to these edges as long as the above condition is satisfied. Therefore, the edges not in T_s need not be part of the model, as long the edges in T_s are tracked, and when assigning weights to the anonymized graph, non-tree edges are assigned weights greater than the shortest path with the largest weight. This captures the information as modeled by the constraints in (2), without adding to the complexity of the model to be solved by the LP solver. Thus, Category III inequalities along with some additional information can model Dijkstra's algorithm, and the complexity of the modified model becomes $O(n)$ ($n-1$ to be exact). The asymptotic complexity of the models in this section and in Section III-A are the same: both grow linearly with n (assuming that d is a constant compared to n). But considering the fact that d is generally of the order of 10 or 100 (as shown in our experiments using social network graphs), the model suggested in this section provides 1 to 2 orders of magnitude reduction in the number of inequalities.

TABLE I
SUMMARY OF THE SOCIAL GRAPHS.

Data Set	No. of Vertices	No. of Edges	Avg. Degree
Flickr-user-3	55,803	6,662,377	119.39
LJ-user-3	15,508	384,947	24.82
Orkut-user-3	26,110	899,638	34.46
Youtube-user-3	237,469	2,457,206	10.35

IV. EXPERIMENTAL EVALUATION

In this section, we experimentally evaluate the two models presented in this paper, compare their performance, and validate our analysis. All the algorithms were implemented in Java, and the experiments were run on an Intel Core 2 Quad Q6600 processor operating at a clock speed of 2.4GHz. The machine has 3GB main memory and runs Fedora Core Linux with kernel 2.6.26.6-49.fc8. We used four real social network data sets obtained from the authors of [3]. In our experiments, we used a free open-source LP Solver (*lp_solve 5.5*) [19]. We report the time taken to generate the model, complexity of the model, and the time taken to solve the models. The model is written to disk, and the system solving the model reads the model from disk, and generates the solution, which is then used to anonymize the model. The reported times therefore include the disk access latencies. More experiments can be found in the extended version of the paper [18].

A. Datasets

Mislove et al. [3] crawled a number of social network sites for analyzing the properties of these large social graphs, and have made their data sets publicly available. Their data sets include the graphs for a number of popular social networking sites: **Flickr**, **LiveJournal**, **Orkut**, and **Youtube**. We model the graphs of these networks as directed graphs where edges have positive weights, but the models can be extended for undirected graphs. The published graph data sets are unweighted, but since our model is not dependent on the semantics of the weights or their magnitude, we assign randomly generated weights (real numbers in the range 1 to 100) to the edges of the graph. We used different distributions for assigning edge weights, but no considerable change in

TABLE II

EXPERIMENTAL EVALUATION OF SINGLE SOURCE SHORTEST PATHS TREE.

	Linear Model					Reduced Model		Summary	
	Number Inequalities				Time	Number of	Time	Times Reduction	% Reduction
Data Sets	**Cat I**	**Cat II**	**Cat III**	**Total**	**Taken (s)**	**Inequalities**	**Taken (s)**	**in Complexity**	**in Time**
Flickr-user-3	204,626	6,457,751	55,802	6,718,179	98.81	55,802	2.835	120.39	97.13
LJ-user-3	39,030	345,917	15,507	400,454	4.783	15,507	0.938	25.83	80.39
Orkut-user-3	72,130	827,508	26,109	925,747	15.735	26,109	1.752	35.47	88.87
Youtube-user-3	417,526	2,039,680	237,468	2,694,674	44.943	237,468	8.226	11.35	81.7

complexity was observed. In our experiments, we used *user driven graph structures* where we select a vertex in the graph as the root, and extract the graph induced by the vertices which are within k *degrees of separation* from the root (a vertex v is the first degree connection of the root v_0 if there exists an edge (v_0, v)). We use the *user* suffix for referring to the user data sets, and for our experiments, we consider 3^{rd} degree of separation (e.g., *Orkut-user-3*). Table I summarizes the different graphs in the data set in terms of the number of vertices, number of edges, and average out-degrees.

B. Single source shortest paths

We experimentally evaluate the two models for *single source shortest paths tree*. We compare the **Linear** model to the **Reduced** model in terms of the complexity of the model, and the time taken to build the model and write it to the disk.

Table II provides the result from these experiments along with a detailed breakup of the number of inequalities, as well as the reduction in complexity and time of the **Reduced** model compared to the **Linear** model. For the **Linear** model, the categories of inequalities in Table II correspond to the categories defined in Section III-A. As is evident from Table II, the **Reduced** model provides about $O(d)$ times improvement in complexity of the models for all the graphs. Depending on the graph, the value of d varies, and so does the factor of improvement. For example, for the *Flickr-user-3* data set, d is 119.39, and the complexity of the **Reduced** model is about 120 times less than that of the **Linear** model. The large reduction in the number of inequalities also affects the time for generating the model, since in the **Linear** model, fewer number of inequalities need to be *generated*, and more importantly, fewer number of inequalities need to be *written to the disk*. This is illustrated by the almost 90% improvement in time to generate the **Reduced** model.

V. CONCLUSION

In this paper, we provided a solution for effective anonymization of weighted social network graphs. We first presented an abstract model that can effectively preserve any linear property of edge weights. As a proof of concept, we considered the *shortest paths problem* and showed how off-the-shelf linear programming libraries can be used to effectively anonymize the graphs. We analyze the complexity of the models, and experimentally validate our analysis using real social network data.

ACKNOWLEDGEMENT

The authors would like to thank Divyakant Agrawal, Pamela Bhattacharya, Sayan Ranu, and the anonymous reviewers for their insightful comments on earlier versions of the paper. The authors would also like to thank Alan Mislove for providing the data sets used for the experiments. This work is partially supported by NSF Grant IIS-0744539.

REFERENCES

[1] M. Girvan and M. E. Newman, “Community structure in social and biological networks.” *Proc Natl Acad Sci U S A*, vol. 99, no. 12, pp. 7821–7826, June 2002.

[2] Y.-Y. Ahn, S. Han, H. Kwak, S. Moon, and H. Jeong, “Analysis of topological characteristics of huge online social networking services,” in *WWW*. New York, NY, USA: ACM, 2007, pp. 835–844.

[3] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, “Measurement and analysis of online social networks,” in *IMC*, 2007, pp. 29–42.

[4] S. Hill, F. Provost, and C. Volinsky, “Network-based marketing: Identifying likely adopters via consumer networks,” *Statistical Science*, vol. 22, no. 2, pp. 256–275, 2006.

[5] L. Getoor and C. P. Diehl, “Link mining: a survey,” *SIGKDD Explor. Newsl.*, vol. 7, no. 2, pp. 3–12, 2005.

[6] L. Backstrom, C. Dwork, and J. Kleinberg, “Wherefore Art Thou R3579X?: Anonymized Social Networks, Hidden Patterns, and Structural Steganography,” in *WWW*, 2007, pp. 181–190.

[7] A. Korolova, R. Motwani, S. Nabar, and Y. Xu, “Link Privacy in Social Networks,” in *ICDE*, 2008, pp. 1355–1357.

[8] A. Campan and T. M. Truta, “A Clustering Approach for Data and Structural Anonymity in Social Networks,” in *PinKDD*, 2008, pp. 1–10.

[9] G. Cormode, D. Srivastava, T. Yu, and Q. Zhang, “Anonymizing bipartite graph data using safe groupings,” *Proc. VLDB Endow.*, vol. 1, no. 1, pp. 833–844, 2008.

[10] M. Hay, G. Miklau, D. Jensen, D. Towsley, and P. Weis, “Resisting structural re-identification in anonymized social networks,” *Proc. VLDB Endow.*, vol. 1, no. 1, pp. 102–114, 2008.

[11] L. Liu, J. Wang, J. Liu, and J. Zhang, “Privacy preservation in social networks with sensitive edge weights,” in *SDM*, 2009, pp. 954–965.

[12] J. M. Kumpula, J. P. Onnela, J. Saramaki, K. Kaski, and J. Kertesz, “Emergence of communities in weighted networks,” *Physical Review Letters*, vol. 99, pp. 228 701–1–228 701–4, 2007.

[13] R. Toivonen, J. M. Kumpula, J. Saramäki, J.-P. Onnela, J. Kertész, and K. Kaski, “The role of edge weights in social networks: modelling structure and dynamics,” *Noise and Stochastics in Complex Systems and Finance*, vol. 6601, no. 1, pp. B1–B8, 2007.

[14] D. Kempe, J. Kleinberg, and E. Tardos, “Maximizing the spread of influence through a social network,” in *KDD*, 2003, pp. 137–146.

[15] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, “Arnetminer: extraction and mining of academic social networks,” in *KDD*. New York, NY, USA: ACM, 2008, pp. 990–998.

[16] J. B. Kruskal, “On the shortest spanning subtree of a graph and the traveling salesman problem,” *Proceedings of the American Mathematical Society*, vol. 7, no. 1, pp. 48–50, February 1956.

[17] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[18] S. Das, Ömer Eğecioğlu, and A. El Abbadi, “Anonymizing Edge-Weighted Social Network Graphs,” Computer Science, UC Santa Barbara, Tech. Rep. CS-2009-03, March 2009.

[19] “LPSolve 5.5,” http://lpsolve.sourceforge.net/5.5/.