Extending [K_1, K_2] Anonymization of Shortest Paths for Social Networks

Yu-Chuan Tsai[1], Shyue-Liang Wang[2(✉)], Tzung-Pei Hong[3], and Hung-Yu Kao[4]

[1] Library and Information Center, National University of Kaohsiung, Kaohsiung 81148, Taiwan
yjtsai@nuk.edu.tw
[2] Department of Information Management, National University of Kaohsiung, Kaohsiung 81148, Taiwan
slwang@nuk.edu.tw
[3] Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung 81148, Taiwan
tphong@nuk.edu.tw
[4] Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan 70101, Taiwan
hykao@mail.ncku.edu.tw

Abstract. Privacy is a great concern when information are published and shared. Privacy-preserving social network data publishing has been studied extensively in recent years. Early works had concentrated on protecting sensitive nodes and links information to prevent privacy breaches. Recent studies start to focus on preserving sensitive edge weight information such as shortest paths. Two types of privacy on sensitive shortest paths have been proposed. One type of privacy tried to add random noise edge weights to the graph but still maintain the same shortest path. The other privacy, *k-shortest path privacy*, minimally perturbed edge weights, so that there exists at least *k* shortest paths. However, there might be insufficient paths that can be modified to the same path length. In this work, we extend previously proposed *[k_1, k_2]-shortest path privacy*, $k_1 \leqq k \leqq k_2$, to not only anonymizing different number of shortest paths for different source and destination vertex pair, but also modifying different types of edges, such as partially visited edges. Numerical experiments showing the characteristics of the proposed algorithm is given. The proposed algorithm is more efficient in running time than the previous work with similar perturbed ratios of edges.

Keywords: Privacy preservation · Shortest path · Anonymity · *k-shortest path privacy* · *[k_1, k_2]-shortest path privacy*

1 Introduction

On-line social networking not only shares information with extreme ease, but also exposes personal identifiable information such as full name, email address, phone numbers, hobbies, and interests to public without notices. Privacy is a great concern

L. Wang et al. (Eds): MISNC 2015, CCIS 540, pp. 187–199, 2015.
DOI: 10.1007/978-3-662-48319-0_15

when information are published and shared. In order to protect personal data from public breaches, national laws on freedom of information and on-line privacy protection acts have been proposed and implemented in many countries [4]. However, while these laws can deter or punish the offenders, but they cannot completely stop the infraction of privacy from publicly available information. Privacy-preserving social network publishing is intended to preserve privacy of individual information on on-line social networking websites and is developed to complement the laws on freedom of information.

A social network is a graph structure made up of entities, the connections between entities and the strength of connections. The entities represent the individuals and the connections between entities represent the relationships or interactions between entities. The edge weights represent the strength of the linked relationships [10], in which the positive edge weights express the trust relation, and the negative edge weights express the bad relation. [14] showed a sensitive shortest path protection example on a weighted social network, which representing an automotive business network between Japanese corporations and American suppliers in North America [9]. The lowest cost path (shortest path) might be desired to be preserved by the Japanese corporation as well, to receive most competitive suppliers. How to protect the sensitive business paths is a great concern in the competitive business environment these days.

In this work, we study the problem of anonymizing the sensitive shortest paths between given pairs of vertices on the weighted directed graph with positive-only edges and positive/negative edge weights. The previous works [19][20], *k-shortest path privacy* and [k_1, k_2]-*shortest path privacy*, made minimal perturbation of edge weights, so that there exists at least k or $k_1 \leqq k \leqq k_2$ shortest paths for all pairs of source and destination vertices respectively. However, it is quite possible that certain sensitive vertex pairs do not have enough paths (or edge weights) to achieve exactly *k-shortest paths privacy* or [k_1, k_2]-*shortest path privacy*. Current work further extends the [k_1, k_2]-*shortest path privacy*, $k_1 \leqq k \leqq k_2$, to modify edges of different types. Based on the greedy approach, we present an algorithm to modify two types of edges, namely None-Visited (*NV*), and partially-visited (*PV*) edges to achieve the [k_1, k_2]-*shortest path privacy*, $k_1 \leqq k \leqq k_2$.

The major contributions are summarized as follows:

- We extend the [k_1, k_2]-*shortest path privacy*, $k_1 \leqq k \leqq k_2$, and propose an algorithm to modify different types of edges, namely none-visited (*NV*) and partially-visited (*PV*), on two types of weighted graph, positive-only and positive/negative edge weight graphs.
- Based on two metrics: running time, and ratio of modified edges, we examined the performance on two types of weighted graphs, positive-only and positive/negative edge weight graphs. Comparison with previous work shows our proposed algorithm is more efficient in running time with similar perturbed ratios of edges.

The rest of the paper is organized as follows. Section 2 describes the related works. Section 3 gives the problem description. Section 4 presents the proposed algorithm. Section 5 reports the numerical experiments. Section 6 concludes the paper.

2 Related Works

Privacy preserving on social networks has concentrated on dealing with re-identification attacks of nodes and links [3][5][6][8][11][12][13][14][24][25]. In order to protect the nodes and link relationships before data is published, most of the techniques modify the graph by adding/deleting nodes/edges. The attackers basically use the background knowledge to attack the published networks, such as passive and active attack [1][2]. The passive attack doesn't change the structure of graph, and the active attack changes the structure of graph.

Weighted graphs can be used for analyzing the formation of communities within the network, business transaction networks, viral and targeted marketing and advertising [6][13][14][18][19][22]. Depending on the applications, the edge weights could be used to represent "degree of friendship", "ratios of opinion", and "business transaction", etc. In addition, edges in a social network could be directed with positive and negative weights that show complex linked relationships [10], with negative links representing the relationships as 'dislike", "distrust", "foes", and so on.

In order to protect the privacy of sensitive edges, four types of works have been proposed on weighted graphs. The first type of works tries to protect the shortest path characteristic between pairs of source and destination vertices. The shortest path remains to be the shortest path after all edge weights are minimally modified [6][14]. The second type of works tries to preserve the privacy of the weights of edges emitting from a given vertex within a predefined parameter, called *k-anonymous weight privacy* [13]. The third type of works studies the shortest distance computing in the cloud which aims at preventing outsourced graphs from neighborhood attacks [7]. The adversary in an outsourced graph cannot calculate the shortest path or shortest distance between neighboring nodes. The fourth type of works studied the *k-shortest path privacy* anonymization on social networks [19][20]. It extends *k-anonymity* concept on relational data to graph data and minimally perturbed edge weights so that there exists at least *k* shortest paths. There are other works on weighted graphs [15][21] that preserve the node identities. Liu et al. [15] proposed a generalization based anonymization approach to achieve *k-possible anonymity*, which used the edges generalization to achieving generalized anonymization groups, on weighted social networks. Yuan et al. [21] proposed a *k-weighted-degree anonymous* model and it prevented the attacks using the node's degree and weight information on the edges adjacent to the nodes as the background knowledge.

For the first two types of works that preserve the shortest paths between pairs of vertices, Gaussian randomization perturbation and greedy perturbation techniques that minimally modify the edge weights without adding or deleting any vertices and edges have been proposed [14]. A linear programming abstract model that can preserve linear properties of edge weights (including shortest paths) after anonymization is presented in [6]. These works do not change the property of the selected shortest path on the anonymized graph. Our work is similar to fourth type of works, but is an extension and more flexible. In our work, we achieve the shortest path privacy for sensitive vertex pairs by modifying two different types of edges simultaneously.

3 Problem Description

In this work, we study the problem of how to flexibly achieve anonymization of sensitive shortest paths between specified source and destination vertices on directed weighted graphs. An information network is represented as a graph $G=(V, E, W)$, where $V=\{v_1, v_2, \ldots, v_n\}$ represents a set of entities, and $E=\{e_{1i}, e_{2i}, \ldots, e_{nn}\}$ represents relationship between entities and $W=\{w_{1i}, w_{2i}, \ldots, w_{nn}\}$ represents strength of the relationships, which could be positive or negative.

For given target source and destination vertex pairs in H, the edges on the shortest paths may overlap with each other. Three types of edges can be classified according to their involvement in the shortest paths on a weighted graph [13][19]: *None-Visited* (*NV*) edges, *Partially-Visited* (*PV*) edges, and *All-Visited* (*AV*) edges. An edge $e_{i,j}$ is a *None-Visited* (*NV*) edge if the edge $e_{i,j}$ does not belong to any shortest path to be preserved. An edge $e_{i,j}$ is a *Partially-Visited* (*PV*) edge if some shortest paths (including those modified), but not all shortest paths, pass through the edge. An edge $e_{i,j}$ is an *All-Visited* (*AV*) edge if all shortest paths (including those modified) from all pairs of source and destination vertices pass through the edge. In this work, we use the definition in [19] and consider two types of edges, *NV* and *PV* edges, and each edge can be anonymized only once.

Figure 1 shows a weighted graph with seven vertices. There are two sensitive shortest paths, SP_1 and SP_2, for two specified sensitive vertex pairs in $H = \{(v_3, v_5), (v_2, v_6)\}$ respectively. The first shortest path, SP_1, is between v_3 and v_5, $\{e_{3,2}, e_{2,5}\}$. The second shortest path, SP_2, is between v_2 and v_6, $\{e_{2,5}, e_{5,6}\}$. The edges $e_{1,2}$, $e_{1,3}$, $e_{2,7}$, $e_{3,5}$, $e_{4,5}$, $e_{4,6}$ are *NV* edges in the initial state, in which they are not passed by any of the shortest paths SP_1 and SP_2 from target node pairs in H. The *PV* edges are those edges that are passed through by only one of the shortest paths SP_1 or SP_2, but not both, such as $e_{3,2}$, $e_{5,6}$, in the initial state. In this example, there is only one *AV* edge, $e_{2,5}$, as it is on both shortest paths SP_1 and SP_2. In practices, the *NV* edges could be the majority edge on a graph. Intuitively, *AV* edges could be rare in a graph. There are more *NV* edges than *PV* edges and more *PV* edges than *AV* edges. In addition, modifying the weights of *NV* edges only change the length of specific path, and modifying the weights of *PV* edges have side effects on other already anonymized shortest paths and perhaps new paths to be anonymized. In addition, the modified edges of each path might be overlapped and required the cyclic check process [18].

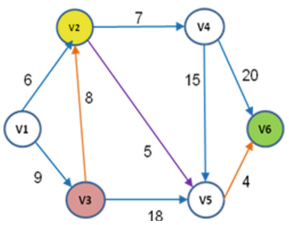

Fig. 1. The Original Network G

For previously proposed k-shortest path privacy [19] and with k=4 on Figure 1, there are only three paths each for both of two sensitive node pairs and there is no available *NV* edges to be modified for second sensitive vertex pair on its second shortest path. In such case, *k-shortest path privacy* cannot be achieved, unless artificial edges or vertices are added so that more paths become available for modification. For previous work [20] of [k_1, k_2]-*shortest path privacy*, when $2 \leq k \leq 4$ on Figure 1, both of these two nodes pairs have two paths respectively. However, there are no *NV* edges available to be modified.

As such, in this work, we propose to extend the [k_1, k_2]-*shortest path privacy*, in previous work, so that, when modifying edge weights, *NV* edges and *PV* edges can be considered simultaneously. The [k_1, k_2]-*shortest path privacy* and its privacy value are given as follows.

Definition 1. ([k_1, k_2]-*shortest path privacy*)

Given a graph G, a set of source and destination vertices H, a privacy level k, the graph G satisfies [k_1, k_2]-shortest path privacy if there exists k_i shortest paths, $k_1 \leqq k_i \leqq k_2$, for i-th vertex pair specified in H.*

According to the definition of [k_1, k_2]-*shortest path privacy*, $k_1 \leqq k \leqq k_2$, there might exist different numbers of anonymized paths for different source and destination vertex pair, so that we use the privacy value to evaluate the level of privacy as follows.

Definition 2. (*Privacy value of* [k_1, k_2]-*shortest path privacy*)

Given a graph G, a set of source and destination vertices H, a privacy level k, the privacy value of an anonymized graph G that satisfies [k_1, k_2]-shortest path privacy is defined as:*

$$privacy \quad value = \frac{1}{n} \sum_i \frac{1}{k_i}$$

The [k_1, k_2]-*shortest path privacy*, $k_1 \leqq k \leqq k_2$, can be applied to multiple sensitive vertex pairs on both weighted un-directed/directed graphs, and prevent the adversary to infer the true sensitive shortest path relationship between any vertex pair. In other words, we try to hide the true sensitive information by cloaking with other shortest paths, depending on how many paths exist between the given vertex pair.

According to the definition, the higher k value will result in lower privacy value, which implies more private and secure. If there is less than k shortest paths for certain vertex pair, compared to *k-shortest path privacy*, [k_1, k_2]-*shortest path privacy* can be anonymized, but *k-shortest path privacy* could not be anonymized by its definition.

4 Proposed Algorithm

This section presents an algorithm, *EKMP*, to achieve the [k_1, k_2]-*shortest path privacy*, $k_1 \leqq k \leqq k_2$, by considering and modifying two types of edges, *NV* and *PV* at the same time. The proposed greedy-based algorithm try to modify the *NV* and *PV* edge weights from the top k, $k_1 \leqq k \leqq k_2$, shortest paths so that they all possess the same path length after modification. We use the weighed-proportional-based strategy to modify

the edge weights. For a path to be anonymized, reducing the edge weights of *NV* edges make the path length equal to the length of the shortest path, the path is anonymized. If it does not have *NV* edges, then will also consider *PV* edges. In addition, the modified edges of already anonymized paths might be overlapped and required the cyclic check process [18]. Due to some paths weights cannot be further reduced, the next longer path will be considered. The algorithm will return the k anonymized paths, $k_1 \leqq k \leqq k_2$. The algorithm will not try to modify the structure of the graph.

As an example, for k_1=2 and k_2=4, Figure 2 shows the anonymized result from Figure 1. There are two sensitive shortest paths, SP_1 and SP_2, for the two target vertex pairs in $H = \{(v_3, v_5), (v_2, v_6)\}$ respectively in Figure 1. The first shortest path, SP_1, is between v_3 and v_5, $\{e_{3,2}, e_{2,5}\}$. The second shortest path, SP_2, is between v_2 and v_6, $\{e_{2,5}, e_{5,6}\}$. Their shortest path lengths are 13 and 9, respectively. For first target vertex pair v_3 and v_5, the second shortest path is P_{12}= $\{e_{3,5}\}$ and the path length is 18. The difference between SP_1 and P_{12} is 5. The *NV* edges, $e_{3,5}$ is modified to 13. The third shortest path is $P_{13} = \{e_{3,2}, e_{2,4}, e_{4,5}\}$ and the path length is 30. The difference between SP_1 and P_{13} is 17. The *NV* edges are $e_{2,4}$ and $e_{4,5}$, and they are modified to 1.6 and 3.4, respectively. In first vertex pair, it only has two other paths. For second target vertex pair v_2 and v_6, the second shortest path is $P_{22} = \{e_{2,4}, e_{4,5}, e_{5,6}\}$ and the updated path lengths is 9. The difference between SP_2 and P_{22} is 0. So, this path has been done. The third shortest path is $P_{23} = \{e_{2,4}, e_{4,6}\}$ and the updated path length is 21.6. The difference between SP_2 and P_{23} is 12.6. The *NV* edge is $e_{4,6}$ and is modified to 7.4. The second vertex pair also has two other paths to be anonymized. For this example, according to our definition, the privacy value is 1/3.

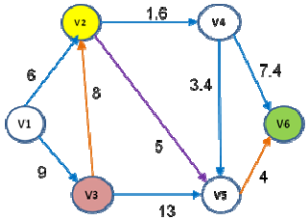

Fig. 2. The [2,4]-shortest Path Privacy Network

The proposed algorithm includes the following major steps.

1. In the initial state, for every vertex pair in H, it finds all paths between the vertices and arranges them in increasing order.
2. Select one target vertex pair and use the weighted-proportional strategy to modify the *NV* edge weights first so that the modified path has the same length as the sensitive target shortest path.
3. If the target path does not find have enough *NV* edges to modify, then we include the *PV* edges in the target path. Double check and update the length of all anonymized paths.
4. Repeat step two to four until all vertex pairs are processed.

Algorithm. [k_1, k_2]-Multiple Paths Anonymization Algorithm (*EKMP*)
Input: W, weighted adjacency matrix of a given graph G,
H, the set of source and destination vertices for which the shortest paths are to be anonymized,
k_1, the minimum number of shortest path between each pair of source and destination vertices,
k_2, the maximum number of shortest path between each pair of source and destination vertices,
Output: anonymized weighted adjacency matrix W*,

for (each distinct start vertices v_i of H)
//find the shortest path and top-k' shortest paths, where $k_1 \leqq k' \leqq k_2$
{*SPL* := *SPL* + shortest path $p_{i,j}$;//*SPL*: shortest path list for (v_i, v_j) in H
DL := *DL* + $d_{i,j}$; //*DL*: shortest paths length list for (v_i, v_j) in H
If ($k_1 \leqq$ |top-k' shortest paths| $\leqq k_2$)
{ *APL* := *APL* + top-k' shortest paths $tp_{i,j}$; //*APL*: top-k' shortest paths list for (v_i, v_j) in H order by path length increasing
PDL := *PDL* + top-k' shortest path lengths $td_{i,j}$; //*PDL*: top-k_2 shortest path lengths list for each pair (v_i, v_j) in H order by path length increasing
k_p := |top-k' shortest paths|;
}
Else { //|top-k' shortest paths|>k_2 or |top-k' shortest paths|<k_1
If (|top-k' shortest paths|>k_2)
{ *APL* := *APL* + top-k_2 shortest paths $tp_{i,j}$; //*APL*: top-k' shortest paths list for (v_i, v_j) in H order by path length increasing, where $k'=k_2$
PDL := *PDL* + top-k_2 shortest path lengths $td_{i,j}$;
//*PDL*: top-k' shortest path lengths list for each pair (v_i, v_j) in H order by path length increasing, where $k'=k_2$
k_p := |top-k' shortest paths|;
}
Else { continue; }
}//end of for, finding the shortest paths and top-k' shortest paths in H, where $k_1 \leqq k' \leqq k_2$
while ($H \neq \phi$) {
$d_{r,s} := \min_H d_{i,j}$; //minimum of all shortest paths
$H := H - (v_r, v_s)$;
while (|*TSPL*| < k_p) { //*TSPL*: there are at most k paths for current vertex pair
$d'_{r,s}$:= pop up first of $PDL_{r,s}$;
p'r,s := pop up first of $APL_{r,s}$;
If ($d'_{r,s} = d_{r,s}$) { //same length
TSPL := *TSPL* + $p'_{r,s}$; // add to anonymized list
continue; } //to find next shortest path
Else { // different length
If ($d'_{r,s}$ and $d_{r,s}$ satisfy cyclic check) then continue;
// $d'_{r,s}$ cannot be anonymized
let *diff* := $d'_{r,s} - d_{r,s}$; //the weight to be reduced
modified_Process($p'_{r,s}$, *diff*) ;
//call a modifying procedure to do the weighted-proportional-process
};
//end of else, different length
}; // end of while (|*TSPL*|< k)
SPL := *SPL* + *TSPL*;
}; // end of while ($H \neq \phi$)

modified_Process($p'_{r,s}$, *diff*)
ML = *NVL* := $p'_{r,s}$ – {edges in *SPL* and *TSPL*}; //*NVL*: the *NV* edges list
If (*ML* ≠ ϕ) { //exist *NV* edges to be modified
for (each edge $(eML)_{i,j}$ on the *ML*) { //reduce proportionally

$$w'_{i,j} = w_{i,j} - \frac{w_{i,j}}{\sum w_{i,j}} \times diff \;;$$

update the adjacency matrix;
TSPL := *TSPL* + $p'_{r,s}$; //save the modified path
}; // end of for each edge $(eML)_{i,j}$
} ; // end of if, check *NV* edges to be modified
Else { //Use the *PV* edges to be modified
PV_Process(); // end of check *PV* edges to be modified
}; //end of if , check *NV* edges to be modified

PV_Process ($p'_{r,s}$, *diff*)
PVL := edges in *SPL* – edges in *TSPL*; //find the *PV* edges that are not used;
PML := *PVL* ∩ $p'_{r,s}$; //consider the *PV* edges that are not modified in the selected path
If (*PML* ≠ ϕ) { // Use *PV* edges to be modified
for (each edge $(eML)_{i,j}$ on the *PML*) { // modified proportionally

$$w'_{i,j} = w_{i,j} - \frac{w_{i,j}}{\sum w_{i,j}} \times diff \;;$$

update the adjacency matrix;
TSPL := *TSPL* + $p'_{r,s}$; //save the modified path
}; // end of for each edge $(eML)_{i,j}$
Update the path lengths in *TSPL* and new length $uad_{i,j}$;
Update the path lengths in *PDL* and *DL*;
for (each path $AP_{i,j}$ in (*TSPL* ∪ *SPL*))
{ // updated the path lengths which had be reduced
let *PVdiff* = $d_{i,j} - uad_{i,j}$;
If (*PVdiff* > 0) { // re-modified the *NV* edges
RML := edge $(eRML)_{i,j}$ on the path $AP_{i,j}$ in (*TSPL*+*SPL*-*H*);
for (each edge $(eRML)_{i,j}$ on the *RML*) {

$$w''_{i,j} = w'_{i,j} - \frac{w'_{i,j}}{\sum w'_{i,j}} \times PVdiff \;;$$

update the adjacency matrix;
}; //end of for each edge $(eRML)_{i,j}$
}; //end of if, check the *PVdiff*
}; //end of for each $AP_{i,j}$
}; // check *PV* edges to be modified

In lines 1 to 16 of the *EKMP* algorithm, it first finds all the shortest paths and top-*k'* shortest paths for all source and destination vertices in *H*, where $k_1 \leqq k' \leqq k_2$. In lines 17 to 33, it performs the anonymization process using the first path of *APL* in increasing order of the path lengths for the selected vertices pair (v_r, v_s). In the *modified_Process()* procedure, it applies the weighted-proportional strategy to modify the edges weights for *NV* and *PV* edges. In the *modified_Process()* procedure, it applied the *NV* edges to modify the edges weights firstly. If certain path does not have any *NV* edges, then it called *PV_Process()* procedure to process this situation. In the *PV_Process()*, it applied the *PV* edges firstly in line 1 to 10, and after modifying *PV*

edges, due to the side effect, it only modifies the *NV* edges in lines 11 to 21. The *PV* edges are only used one time.

5 Numerical Experiments

To evaluate the performance of the proposed algorithm, we run simulations on a real world dataset, *hep-th* [16]. In the experiments, we randomly generate four target vertex pairs and construct sets of vertex pairs H for $|H|$=2, 3, respectively. All experiments reported in here were carried out on Intel core i7 CPU, 2.67GHz machine with 4GB RAM, running Microsoft Windows 7 operating system. The algorithm was implemented in Microsoft Visual Studio 2005. We fixed the k_1, k_1=2, and varied the k_2 values of anonymity in the range of 3 and 20.

The following experimental results demonstrate the performance of proposed algorithm. We evaluated the running times, and the ratios of modified edges of our proposed algorithm on two types of graphs: positive/negative edge weight graph and positive-only edge weight graph. We also compare with previous work in [20], namely *K1K2MPN* on both positive and negative edges and, *K1K2MPP* on positive-only edges, with two metrics: running time and the ratio of modified edges.

A. Dataset

To demonstrate the characteristics and evaluate the performance of proposed algorithm, we run simulations on one real world dataset, *hep-th*. The *hep-th* dataset was a weighted network that describes a co-authorship network of high energy physics scientists, which was compiled by M. Newman in 2001 [16]. The hep-th dataset contains 7,610 scientists (nodes) and 15,751 co-authorships (edges), and each edge is assigned with a real value weight.

B. Performance Analysis

We examine the performance of our proposed algorithm with two metrics: running times, ratios of modified edges. The running time indicates the computation efficiency of the algorithm. The ratios of modified edges indicate the percentage of edges affected and modified by the algorithm. It is defined as the number of modified edges over the total number of edges on the k' shortest paths, $k_1 \leqq k \leqq k_2$, for vertices specified in H. In the experiments, our algorithm run simulation on two types of graphs, the first type allows both positive and negative edge weights, namely *EKMPN*, and the second type allows positive-only edges, namely *EKMPP*.

C. Discussion

To compare the performance of *EKMPN* and *EKMPP*, Figure 3 shows the results of running times on different types of graphs for different sizes of H. On positive-edge-weight-only graphs, there exist paths that cannot be modified to the same length as

shortest path. As such, more paths need to be checked and modified and therefore take longer running time.

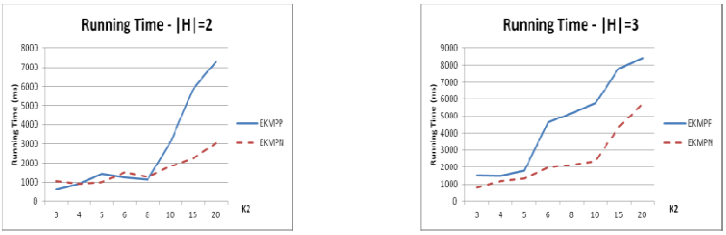

Fig. 3. Comparison of running times with different sizes of H.

Figure 4 shows the ratios of modified edges on different types of graphs with different sizes of H on varies k_2 values. Both types of graphs have similar ratios of modified edges, the *EKMPN* modified less number of edges than *EKMPP*.

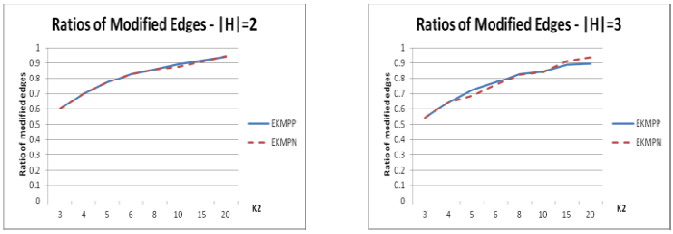

Fig. 4. Comparison of the ratios of modified edges with different sizes of H.

As shown in Figure 5, the proposed algorithms, *EKMPN* and *EKMPP*, have faster running time than previous work, *K1K2MPN* and *K1K2MPP* (modify *NV* edges only) with two different size of *H*. *EKMPN* has better performance then *EKMPP* on running time with two different size of *H*. In Figure 6, *EKMPN* and *EKMPP* have similar ratios of modified edges compared to *K1K2MPN* and *K1K2MPP*. Hence, our proposed algorithms have better performance than the previous work.

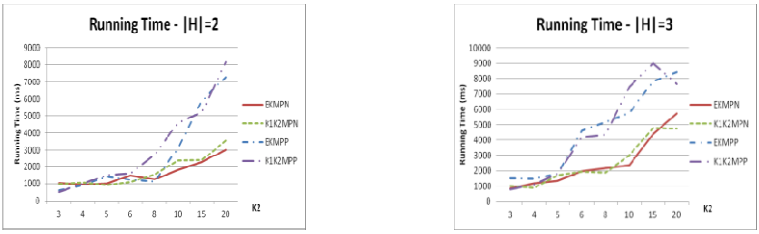

Fig. 5. Comparison of running time with different sizes of H for different methods.

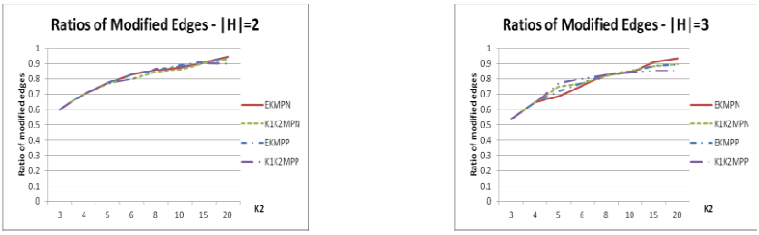

Fig. 6. Comparison of ratios of modified edges with different sizes of H for different methods.

In summary, the proposed *EKMPN* on positive/negative edge weight graphs has better performance than *EKMPP* on positive-only edge weight graphs. In addition, *EKMPN* and *EKMPP* have faster running times than *K1K2MPN* and K1K2MPP, but with similar ratios of modified edges.

6 Conclusion

In this work, we have studied the problem of anonymizing sensitive shortest paths on information networks. We extended the [k_1, k_2]-*shortest path privacy*, $k_1 \leqq k \leqq k_2$, and presented an algorithm based on the greedy approach to modify two different types of edges, namely *NV* and *PV* edges at the same time. We used the privacy value (the equation) to quantify the privacy level. Numerical experiments examining the characteristics of the proposed algorithms were given. The results demonstrated that the proposed algorithm is feasible to achieve the [k_1, k_2]-*shortest path privacy*, $k_1 \leqq k \leqq k_2$, with different performances on both types of graphs and more efficient than previous approach.

In the future, we will consider how to increase the data utility of anonymized graph. In addition, different approaches of modification should be considered, such as adding/deleting nodes/edges and preserving other types of sensitive characteristics such as k-degree for each node on the shortest paths.

Acknowledgment. This work was supported in part by the National Science Council, Taiwan, under grant NSC-101-2221-E-390-030-028-MY3.

References

1. Backstrom, L., Huttenlocher, D.P., Kleinberg, J.M., Lan, X.: Group formation in large social networks: membership, growth, and evolution. In: Proceedings of KDD, pp. 44–54 (2006)
2. Backstrom, L., Dwork, C., Kleinberg, J.M.: Wherefore art thou r3579x?: anonymized social networks, hidden patterns. and structural steganography. In: Proceedings of World Wide Web, pp. 181–190 (2007)
3. Bhagat, S., Cormode, G., Krishnamurthy, B., Srivastava, D.: Class-based graph anonymization for social network data. In: Proceedings of Very Large Data Bases, pp. 766–777 (2009)

4. Banisar, D.: Freedom of information around the world 2006 – A global survey of access to government information Laws. www.privacyinternational.org/foi/survey
5. Cheng, J., Fu, A., Liu, J.: K-isomorphism: privacy preserving network publication against structural attacks. In: Proceedings of ACM SIGMOD Conference, pp. 459–470 (2010)
6. Das, S., Egecioglu, O., Abbad, A.E.: Anonymizing weighted social network graphs. In: Proceedings of International Conference on Data Engineering, pp. 904–907 (2010)
7. Gao, J., Xu, Y., Jin, R.M., Zhou, J.S., Wang, T.J., Yang, D.Q.: Neighborhood-privacy protected shortest distance computing in cloud. In: Proceedings of ACM SIGMOD Conference, pp. 409–420 (2011)
8. Hay, M., Miklau, G., Jensen, D., Towsley, D.F., Weis, P.: Resisting structural re-identification in anonymized social networks. Proceedings of the VLDB Endowment **1**(1), 102–114 (2008)
9. Inkpen, A.: The Japanese corporate network transferred to North America: implications of North American firms. The International Executive **36**(4), 411–433 (1994)
10. Leskovec, J., Huttenlocher, D., Kleingerg, J.: Signed networks in social media. In: Proceedings of CHI Conference on Human Factors in Computing Systems, pp. 1361–1370 (2010)
11. Li, Y., Shenm, H.: On Identity Disclosure in Weighted Graphs. In: Proceedings of the 11th International Conference on Parallel and Distributed Computing, Applications and Technologies, pp.166–174 (2010)
12. Liu, K., Terzi, E.: Towards identity anonymization on graphs. In: Proceedings of ACM SIGMOD Conference, pp. 93–106 (2008)
13. Liu, L., Liu, J., Zhang, J.: Privacy preservation of affinities in social networks. In: Proceedings of International Conference on Information Systems (2010)
14. Liu, L., Wang, J., Liu, J., Zhang, J.: Privacy preservation in social networks with sensitive edge weights. In: Proceedings of the SIAM International Conference on Data Mining, pp. 954–965 (2009)
15. Liu, X., Yang, X.: A Generalization Based Approach for Anonymizing Weighted Social Network Graphs. In: Wang, H., Li, S., Oyama, S., Hu, X., Qian, T. (eds.) WAIM 2011. LNCS, vol. 6897, pp. 118–130. Springer, Heidelberg (2011)
16. Newman, M.E.J.: The structure of scientific collaboration networks. In: Proceedings of Natl. Acad. Sci. **98**, pp. 404409 (2001)
17. Nobari, S., Karras, P., Pang, H., Bressan, S.: L-opacity: linkage-aware graph anonymization. In: Proceedings of 17th International Conference on Extending Database Technology, pp. 583–594 (2014)
18. Wang, S.-L., Tsai, Z.-Z., Hong, T.-P., Ting, I.-H.: Anonymizing Shortest Paths on Social Network Graphs. In: Nguyen, N.T., Kim, C.-G., Janiak, A. (eds.) ACIIDS 2011, Part I. LNCS, vol. 6591, pp. 129–136. Springer, Heidelberg (2011)
19. Tsai, Y.C., Wang, S.L., Kao, H.Y., Hong, T.P.: Edge types vs privacy in *K*-anonymization of shortest paths. In Applied Soft Computing **31**, 348–359 (2015)
20. Tsai, Y.C., Wang, S.L., Kao, H.Y., Hong, T.P.: $[k_1, k_2]$-anonymization of Shortest Paths, In: Proceedings of 4th International Workshop on Intelligent Data Analysis and Management (2014)
21. Yuan, M., Chen, L., Yu, P.S.: Personalized privacy protection in social networks. In: Proceedings of the 36rd International Conference on Very Large Data Bases, pp.141–150 (2010)
22. Yuan, M., Chen, L.: Node Protection in Weighted Social Networks. In: Yu, J.X., Kim, M.H., Unland, R. (eds.) DASFAA 2011, Part I. LNCS, vol. 6587, pp. 123–137. Springer, Heidelberg (2011)

23. Yen, J.Y.: A shortest path algorithm, Ph.D. dissertation, University of California, Berkeley (1970)
24. Zhou, B., Pei, J.: Preserving privacy in social networks against neighborhood attacks. In: Proceedings of the 24th International Conference on Data Engineering, pp. 506–515 (2008)
25. Zou, L., Chen, L., Ozsu, M.T.: K-automorphism: A general framework for privacy preserving network publication. In: Proceedings of the 35rd International Conference on Very Large Data Bases, pp. 946–957 (2009)