An Iterative Algorithm for Graph De-anonymization

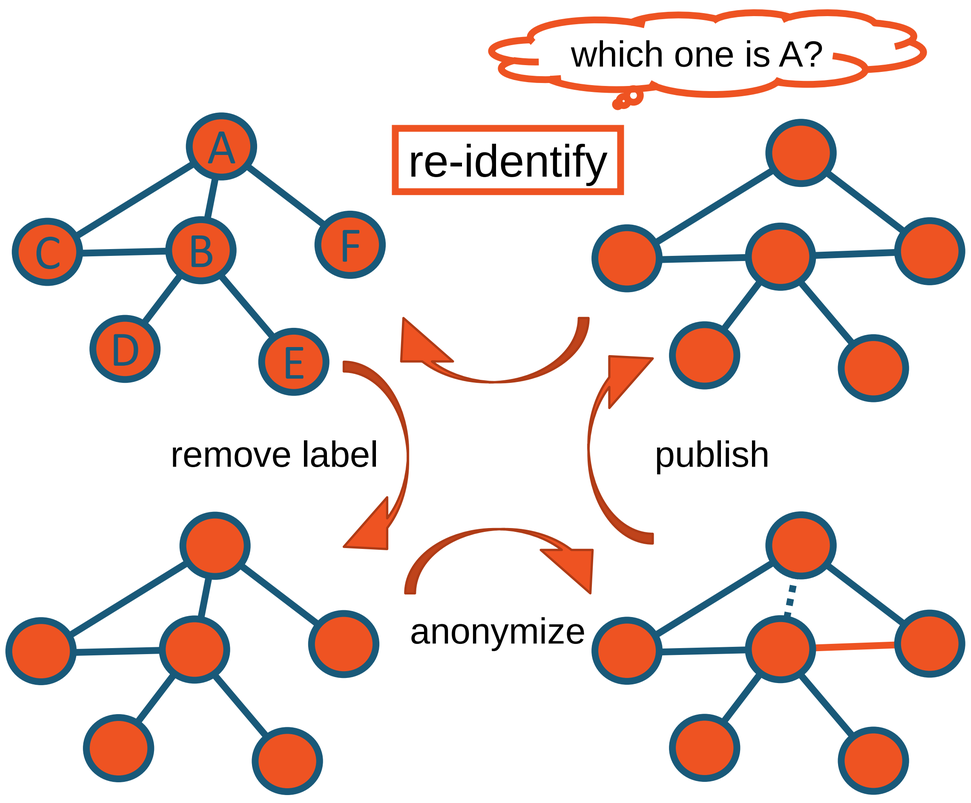
Jun Zhang[1] Youze Tang[1] Xiaokui Xiao[1] Yin Yang[2] Zhenjie Zhang[2] Marianne Winslett[2,3]

1.Nanyang Technological University
Singapore
{jzhang027, s110013, xkxiao}@ntu.edu.sg

2. Advanced Digital Sciences Center
Singapore
{yin.yang, zhenjie}@adsc.com.sg

3. University of Illinois
USA
winslett@illinois.edu

1. Problem Statement



2. Anonymization Techniques

(i) Liu and Terzi's approach: *k-degree anonymization*. Adding edges to original graph in order to ensure that any node in anonymized graph G^* has the same degree with at least $k-1$ other nodes in G^*.

(ii) Bonchi et al.'s approach: *random perturbation and sparsification*. Removing and/or adding edges to original graph randomly to resist structural re-identification.

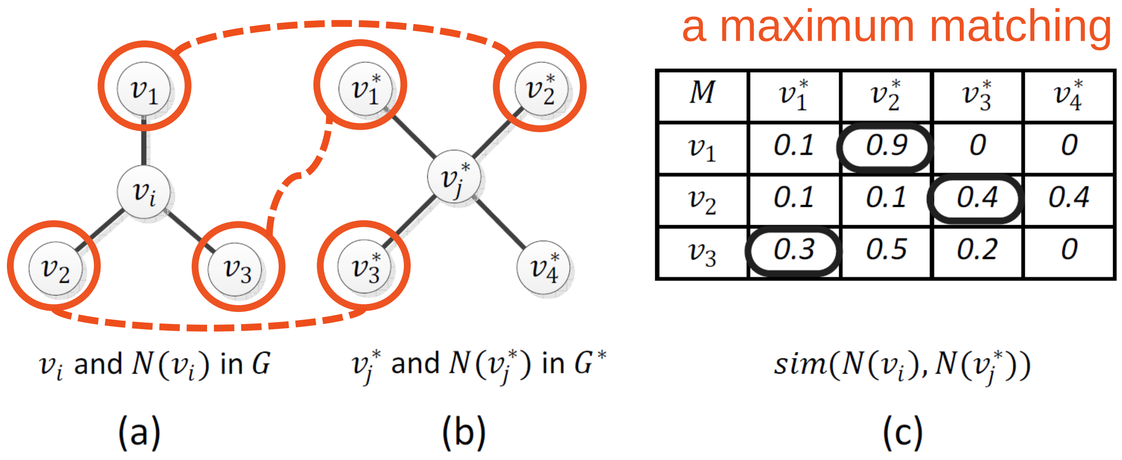
(iii) Tassa and Cohen's approach: *sequential clustering*. Grouping nodes in original graph into clusters, each contains at least k nodes. Next publishing only aggregated information to prevent privacy breach.

3. Solution

Overview: Our algorithm runs in an iterative manner. In particular, it maintains a $n \times n$ *correspondence matrix* M, where the entry e_{ij} at the i-th row and j-th column of M quantifies the similarity between i-th node v_i in original graph G and j-th node v_j^* in anonymized graph G^*. Initially, all entries are set to 1. After that, the algorithm iteratively refines M to adjust the value of each entry until a stop criterion is met.

Node Similarity: Within a certain iteration of our algorithm, we evaluate the similarity between any node v_i in G and any node v_j^* in G^* by inspecting the entry e_{ij}, as well as the neighbors of v_i, denoted by $N(v_i)$, and the neighbors of v_j^*, denoted by $N(v_j^*)$. In particular, we try to figure out the similarity of neighborhoods by matching the nodes in $N(v_i)$ to those in $N(v_j^*)$. More formally, we update e_{ij} with the following equation:

$$e_{ij} = \frac{e_{ij} + sim(N(v_i), N(v_j^*))}{1 + \max(|N(v_i)|, |N(v_j^*)|)}$$



M	v_1^*	v_2^*	v_3^*	v_4^*
v_1	0.1	0.9	0	0
v_2	0.1	0.1	0.4	0.4
v_3	0.3	0.5	0.2	0

Figure 1: An example of sim function

Computation of $sim(N(v_i), N(v_j^*))$:To compute *sim* function, we first construct a complete bipartite graph B consisting of the nodes in $N(v_i)$ and $N(v_j^*)$. For any edge in B connecting v_α in $N(v_i)$ and v_β^* in $N(v_j^*)$, we set the weight of the edge to the value of the entry $e_{\alpha\beta}$ representing the similarity between v_α and v_β^*. After that, we compute a maximum matching in B using the classic Hungarian algorithm, and we set $sim(N(v_i), N(v_j^*))$ to the total weight of the edges in the maximum matching. Intuitively, the larger $sim(N(v_i), N(v_j^*))$ is, the more likely that nodes in $N(v_i)$ can be matched to the nodes in $N(v_j^*)$.

4. Experiment

Dataset: a social network G representing coauthorship of data mining community, with 8,248 nodes and 18,732 edges.

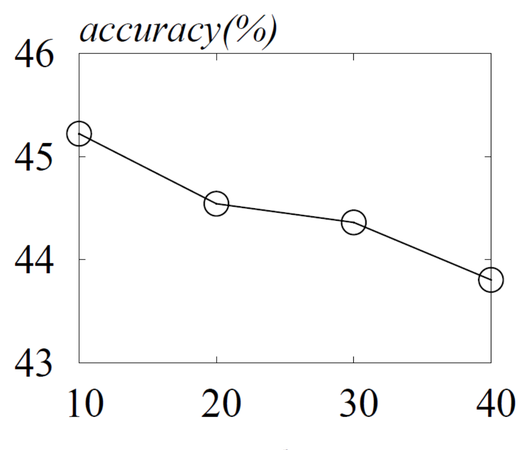


Figure 2: Accuracy against Liu and Terzi's approach

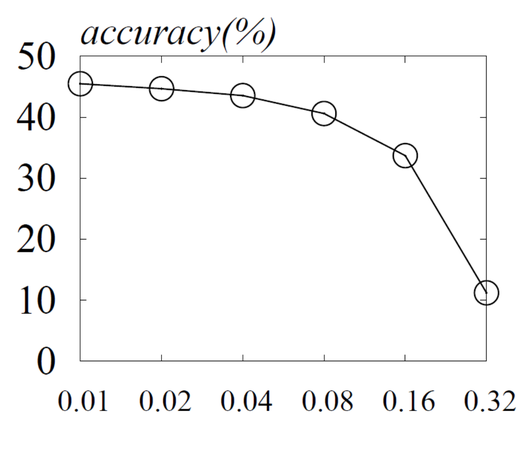


Figure 3: Accuracy against Bonchi et al's sparsification

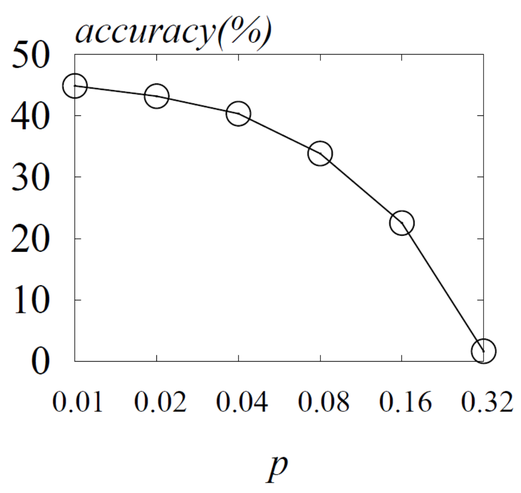


Figure 4: Accuracy against Bonchi et al's perturbation

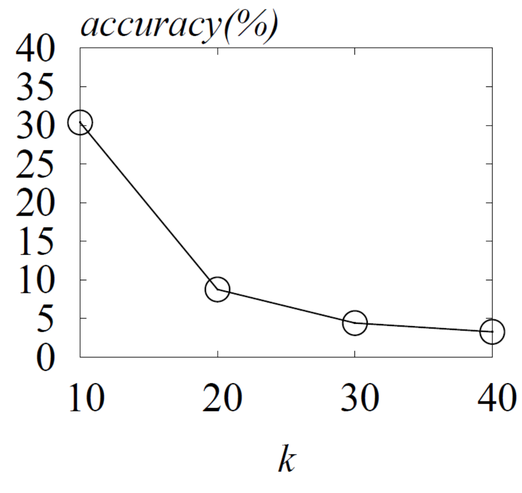


Figure 5: Accuracy against Tassa and Cohen's approach