Fast Multi-hop Neighborhood Structure Matching for Graph De-Anonymization

Aashis Khanal
Department of Computer Science
Georgia State University, Atlanta, Georgia 30303
akhanal1@student.gsu.edu

Abstract

Graph De-anonymization has multiple applications—de-anonymization can help understand relationship of two graphs, it can also help strengthen the privacy of crucial real world graphs like social networks. The general approach to solve this problem is to match neighbourhood structure [1], eccentricity [2]. However, optimal solution with perfect neighborhood matching is non-trivial in the sense that adding few nodes and edges can exponentially increase the solution space. In this paper, we have extended the 1-hop neighborhood matching [1] to K-hop. Naturally, this also adds extra complexity, but we use *Degree Bounded Pre-Node-Filtering* to drastically reduce the complexity than in naive case. The implementation, readme, and the results can be found in **https://github.com/sraashis/graph-de-anonymization**.

I. Introduction

Mapping a node u from one graph G_1 to v of G_2 is not trivial as network structure can be very complex. The complexity is exponentially higher if we have no any initial information(seed mapping). Thus the problem of graph de-anonymization comes in two flavour.

II. Seed based de-anonymization

Seed is a very small number of correct mapping of nodes from G_1 to G_2. The simplest approach to start with the seed information is to keep track of how many neighboring nodes of two unmapped nodes $u \in G_1$, and $v \in G_2$ actually are mapped[1], [2].

$$SIM_1^{sb}(u,v) = \frac{map_1(u,v)}{\sqrt{deg(u)}\sqrt{deg(v)}} \tag{1}$$

, where $map_k(p,q) = \sum_{i=1}^{k} count(p \in nei_i(u) \rightarrow q \in nei_i(v))$, $nei_i(u)$ is all the nodes reachable by $i-hop$ from u.

Matching based on $1-hop$ neighborhood can be ambiguous and misleading in mapping correct nodes as the structural/connectivity identity of a node goes beyond immediate neighborhood. Thus, in this work we have introduced a *K-hop* similarity matching as follows

$$SIM_K^{sb}(u,v) = log(1 + \frac{map_1(u,v)}{\sqrt{nc_1(u)\cdot nc_1(v)}} + \sum_{k=2}^{K} \frac{\prod_{i=1}^{k} map_i(u,v)}{log(\prod_{j=1}^{k-1} nc_j(u)\cdot nc_j(v))\cdot\sqrt{nc_k(u)\cdot nc_k(v)}}) \tag{2}$$

, where $SIM_k^{sb}(u,v)$ is the $seed-based\ K-hop$ similarity between two nodes u, v, and $nc_i(u)$ is number of nodes within $i-$hop from node u. We use $K=3$ based on our resource limitations.

A. Pre-filtering nodes by Degree Bounds

Matching all the nodes of the two graphs takes $O(n^2)$ in terms of complexity. $\forall u \in G_1$, instead of matching to all nodes in G_2, we can match to nodes $v \in G_2$, such that $|deg(u) - deg(v)| \leq D_{lim}$ (We use D_{lim} = 10 initially). The notion is that the likelihood of two nodes with *large degree difference* being matched is very low. For example, a node in G_1 with degree 20, has very low chance of mapping to a node of degree 100 in G_2. However, it has a higher chance of matching to a node in G_2 within degree $|20 \pm D_{lim}|$.

In figure 1, we can see the distributions of nodes within (5-green, 10-yellow, 15-red) degree bounds. It also tells us that the average number of comparison we have to do for each node in G_1 to find a match inG_2—for $D_{lim} = 5$ (green)—around 400 average comparisons has to be done—we needs around 700 comparisons on average for $D_{lim} = 10$ (yellow)—finally, for the

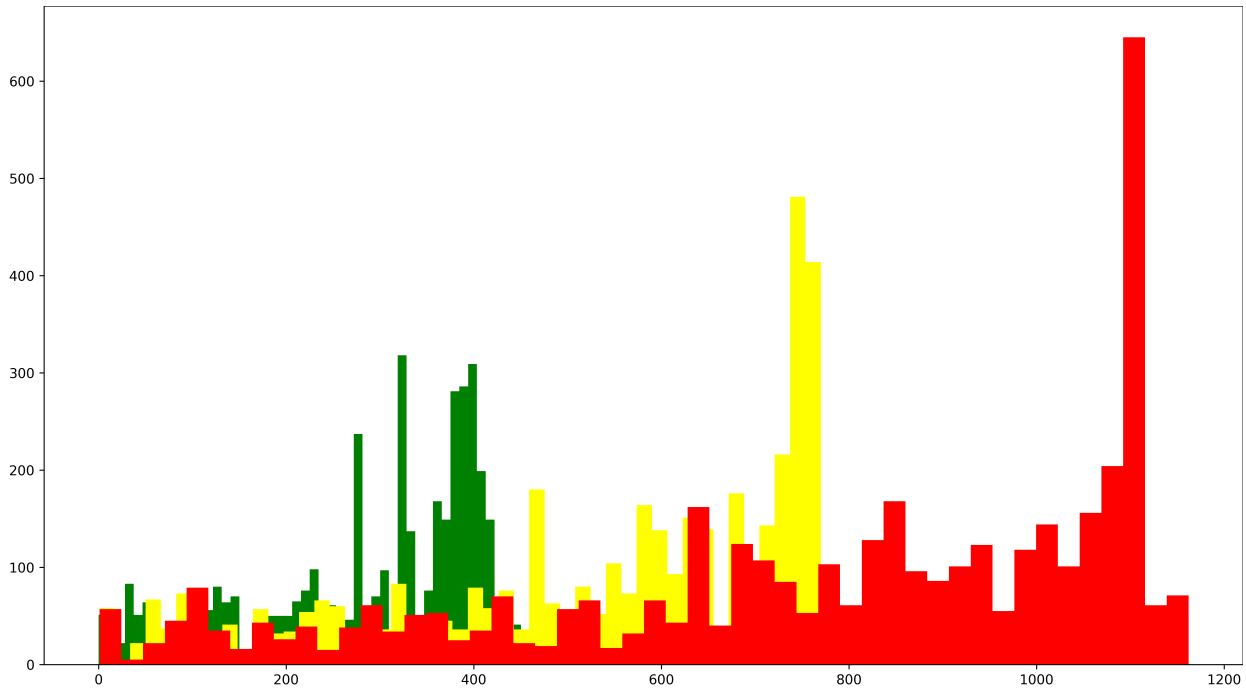

Fig. 1. Number of nodes in G_2 within different degree bound from each nodes of G_1. D_{lim} of 5 for green, 10 for yellow, and 15 for red. It tells us how many average comparisons each node on G_1 has to perform in order to find a match in G_2 using **Pre-Filtering** technique.

$D_{lim} = 15$ (red), we need 1100 comparisions on average, which is significantly more less than the naive comparing $u \in G_1$ to every nodes of g_2.

$MAP =$ Initialize with seed node mapping from $G_1 \rightarrow G_2$ as key, value pairs.

$\mathcal{M} = 500$; Mapping Per Iteration.

Result: Mapping of each nodes from $G_1 \rightarrow G_2$ in MAP

for *i in range(1, 2* $\times \frac{len(G_1)-len(MAP)}{\mathcal{M}}$*)* **do**

  $M = \{\}$;

  for $u \in G_1^{nodes} - MAP_{G_1}^{keys}$) **do**

    $map = \{\}$;

    $D_{lim}^i = D_{lim} \times \sqrt{i}$;

    for $v \in PreFiltered_u(v, G_2^{nodes} - MAP_{G_2}^{values}, D_{lim}^i)$ **do**

      $map[v] = sim_K(u, v)$;

    end

    $M[max_{key}(map)] = max_{value}(map)$;

  end

  Update $top - \mathcal{M}$ mappings from M to MAP;

  if *All nodes from* G_1 *mapped* **then**

    stop;

  end

end

Algorithm 1: Seed based de-anonymization algorithm on multi-hop structure matching

B. Seed Based multi-Hop structure matching De-Anonymization algorithm

The pseudo-code for our technique is listed in algorithm 1. We can see the number of nodes to match decreases linearly based on $\mathcal{M}$ because on each iteration we discard the nodes that we already mapped. We can implement the innermost $for - loop$ in parallel where each threads finds a match of node u in the filtered nodes from G_2 as discussed in section II-A. We ran 32 threads in a Intel Xeon based server with 32 total threads in our experiments. One notable thing is that after using the *Pre-Filtering* technique, the running speed was reduced significantly(Reduced from days to hours). This is natural because most graph are sparse, thus such filtering technique will discard a significant portion of comparisons. We match $\mathcal{M}$ nodes on each iteration making the seed propagation more robust. We can argue that later in the iterations, it would be relatively harder to find a match, thus we increase the D_{lim} space by a factor of $\sqrt{i}$ on every iteration. This has two benefits—1) we can start from a small D_{lim} making the comparison process optimal because initially would have maximum number of nodes from G_1 to be mapped. However, as we move to next iteration, we reduce the node in G_1 to be matched by $\mathcal{M}$. 2) When we have smaller pool of nodes to match, it is wise to have a bigger search space s that our math is near optimal. That's why we increase the node degree bounds as we advance in iterations.

III. Seed free De-Anonymization

Seed free De-Anonymization is more trickier than the seed-based one in the sense that the starting mapping determines how well the later mappings will be. The natural way is to first find few good mappings(can also be called seeds), and then one can use algorithm 1 for the rest of the mapping. However, as mentioned earlier, in order for the algorithm to work, the seed we produce should be near perfect to get a decent result.

A. Seed Extraction with multi-Hop density matching

Following are the some preliminary equations—$nei_i^{'}(u)$ is the nodes that are exactly $i - hop$ distance from u(equation 3), $cn_i^{'}(u)$ is the number of nodes that are exactly $i - hop$ distance from u(equation 4), $ce_i^{'}(u)$ is the number of edges within the sub-graph formed by nodes that are exactly in $i - hop$ distance from u(5).

$$nei_i^{'}(u) = nei_i(u) - \sum_{j=1}^{i-1} nei_j(u) \tag{3}$$

Similarly,

$$cn_i^{'}(u) = cn_i(u) - \sum_{j=1}^{i-1} cn_j(u) \tag{4}$$

$$ce_i^{'}(u) = subgraph(G, nei_i^{'}(u)).numberOfEdges() \tag{5}$$

$$\beta_i(u,v) = \frac{ce_i^{'}(u) \cdot ce_i^{'}(v))}{cn_i^{'}(u) \cdot cn_i^{'}(v)} \tag{6}$$

$$\gamma_1 = \beta_1(u,v) \times |\frac{cn_1^{'}(u)}{ce_1^{'}(u)} - \frac{cn_1^{'}(v)}{ce_1^{'}(v)}| \tag{7}$$

$$\gamma_k = \beta_k(u,v) \times |\frac{cn_k^{'}(u)}{cn_k(u)} - \frac{cn_k^{'}(v)}{cn_k(v)}|,\ for\ k > 1: \tag{8}$$

For, $k = 1$, nodes within 1-hop distance from u, $cn_1(u) = cn_1^{'}(u)$, where $cn_i^{'}(u)$ being the nodes that are exactly $k - hop$ from u, whereas $cn_k(u)$ are the nodes that are within $k - hop$ distance from u. Finally, we can have a numerically stable node similarity metrics for $seed - free$ graph de-anonymization as follows:

$$SIM_K^{SF}(u,v) = \frac{3}{e^{\sum_{i=1}^{K} \gamma_i}} \tag{9}$$

Again, the problem we face is the space complexity—Crude way would be to compare all m nodes from G_1 to n nodes of G_2 making complexity $O(n^2)$. However, we only plan to extract $\mathcal{S}$ (We use $\mathcal{S}$ = 500 for our experiment) nodes which would serve as the seed for algorithm 1. But as mentioned before, we want this seed to be as accurate as possible as our entire downstream mapping depends on it. Thus, we have introduced multi-Hop density matching technique (equation 9) with *degree bounded seed matching*. As such, instead of $O(n^2)$ comparisons, we only do $O(\mathcal{S}.^2)$ We take top-$\mathcal{S}$ nodes from G_1 sorted by degree in descending order. Similarly, we take top-$2 \times \mathcal{S}$ nodes from G_2 after sorting by degree in descending order. This enables to reduce the search space, and the same time, also increase the likelihood of finding the correct match of u in the

selection top-2 $\times \mathcal{S}$ nodes from G_2 by using the similarity metrics in equation 9. Once we have our seed, we can follow the algorithm 1 to find the entire mapping from $G_1 \rightarrow G_2$. We used $\mathcal{M}$ = 1000 for seed free case. The inner loop of this one is also parallelized to get the benefit of multiple threads.

IV. Discussion

The seed extraction technique in **seed-free** method has a subtle limitation-That is when mapping $\mathcal{S}$ nodes from G_1 to same number of nodes in G_2, some seeds(around 64 in our experiment) were mapped to more than one nodes to G_2. This can be easily solved by picking $top\text{-}l$(l=5, 10, ...) nodes on G_2, for each nodes on G_1, and fixing the duplicates by using the SIM_K^{sf} in equation 9. We couldn't encompass this change due to time limitation. However, for **seed-based matching**, our implementation ensure that there are no duplicates mapping by using the matching similarity as in equation 2. For the aforementioned reason, the correctness of mapping for seed-free mapping depends heavily on the quality of seed generated. We believe, this can be improved if we use the previous mentioned technique to avoid duplicate mappings. For the same reason of time limitation, some of the nodes ($< 2\%$ in seed-based, and $< 5\%$ in seed-free) are not matched to any nodes(Hence are mapped to None). We believe running the algorithm for few more iterations would solve that problem. However, because of time constraint, we leave that for future improvements.

References

[1] S. Nilizadeh, A. Kapadia, and Y.-Y. Ahn, "Community-enhanced de-anonymization of online social networks," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 537–548. [Online]. Available: https://doi.org/10.1145/2660267.2660324

[2] A. Narayanan and V. Shmatikov, "De-anonymizing social networks," *CoRR*, vol. abs/0903.3276, 2009. [Online]. Available: http://arxiv.org/abs/0903.3276