Data Protection & Privacy
II Homework: $k - degree$ Anonymity

October 27, 2018

Goal

The goal of this homework is to solve the following problem: given a graph G and an integer k, modify G via a set of edge-addition operations in order to construct a new *k-degree* anonymous graph $\hat{G}$, in which each node v has the same degree of at least $k-1$ other nodes in G [1].

k-*degree* anonymity in a nutshell

Let $G(V, E)$ be an undirected graph; V is a set of nodes and E the set of edges in G. Let $\boldsymbol{d}_G$ be a vector of size $n = |V|$ such that $\boldsymbol{d}_G(i)$ is the degree of the i-th node of G. Without loss of generality, it is also assumed that entries in $\mathbf{d}$ are in decreasing order, such that $\boldsymbol{d}(1) \geq \boldsymbol{d}(2) \geq \ldots \geq \boldsymbol{d}(n)$. Additionally, for $i < j$ we use $\boldsymbol{d}[i, j]$ to denote the subsequence of $\boldsymbol{d}$ that contains elements $i, i+1, \ldots, j-1, j$.

A graph $G(V, E)$ is k-degree anonymous if the degree sequence of G, $\boldsymbol{d}_G$, is k-anonymous. A vector of integer $\boldsymbol{d}_G$ is $k-$anonymous, if every distinct value in $\boldsymbol{d}_G$ appers at least k times.

The Graph Anonymization problem

Given a graph $G(V, E)$ and an integer k, build a $k-$degree anonymous graph $\hat{G}(V, \hat{E})$ with $\hat{E} \cap E = E$ (or $\hat{E} \cap E \approx E$[1] in relaxed form) such that $G_A(\hat{G}, G)$ is minimized, where $G_A(\hat{G}, G) = |\hat{E}| - |E|$.

In the above formulation we want to find the $k-$degree anonymous graph that incurs the minimum graph anonymization *cost*, that is, we want to add the minimum number of edges to the original graph to obtain a $k-$degree anonymous version of it. Such result can be achieved by minimizing the L_1 distance of the degree sequence of G and $\hat{G}$

$$L_1(\hat{\boldsymbol{d}}, \boldsymbol{d}) = \textstyle\sum_i |\hat{\boldsymbol{d}}(i) - \boldsymbol{d}(i)|$$

This is due to the fact that

$$G_A(\hat{G}, G) = |\hat{E}| - |\hat{E}| = \tfrac{1}{2} L_1(\hat{\boldsymbol{d}} - \boldsymbol{d})$$

[1] i,e,. applying only edge additions

A "greedy" algorithm

The algorithm is divided into two steps:

1) Starting from $\boldsymbol{d}$, we construct a new degree sequence $\hat{\boldsymbol{d}}$ that is k-anonymous such that the *degree-anonymization* cost

$$D_A(\hat{\boldsymbol{d}}, \boldsymbol{d}) = L_1(\hat{\boldsymbol{d}} - \mathbf{d})$$

is minimized.

2) Given the new degree sequence $\hat{\boldsymbol{d}}$, we then construct a graph $\hat{G}(V, \hat{E})$ such that $\boldsymbol{d}_{\hat{G}} = \hat{\boldsymbol{d}}$ and $\hat{E} \cap E = E$ (or $\hat{E} \cap E \approx E$ in relaxed form).

The proposed algorithm is `greedy`: it first builds a group made by the first k highest-degree nodes and assigns to each of them a degree equal to $\boldsymbol{d}(1)$[2]. Then it checks whether it must merge the $(k+1)^{th}$ node into the previously formed group or start a new group at position (k+1). In order to take such decision, the algorithm calculates the following two cost values:

$$C_{merge} = (\boldsymbol{d}(1) - \boldsymbol{d}(k+1)) + I(\boldsymbol{d}[k+2, 2k+1])$$

and

$$C_{new} = I(\boldsymbol{d}[k+1, 2k])$$

where

$$I(\boldsymbol{d}[i, j]) = \textstyle\sum_{l=1}^{j}(\boldsymbol{d}(i) - \boldsymbol{d}(l))$$

If $C_{merge} > C_{new}$, a new group is built starting with the $(k+1)$-th node. Then, the algorithm continues recursively for the sequence $\boldsymbol{d}[k+1, n]$. Otherwise, the $(k+1)^{th}$ node is merged to the previous group and the $(k+2)^{th}$ node is considered for merging or as a starting point of a new group. The algorithm terminates after considering all n nodes.

Graph Construction: in this step we use the `ConstructGraph` algorithm visible in Fig 1 to build the anonymized graph.

Dataset

In this homework, datasets can be created through the script `create_graph.py` (written in python). Such script creates a database with a list of English surnames (`engwales_surname.csv`).
To create a simple graph run the script in this way:

```
python create_graph.py max_node min_edge max_edge csv_name_file
```

[2] i.e., the highest degree, remember that the entries in $\boldsymbol{d}$ are ordered in decreasing order.

Algorithm 1 The `ConstructGraph` algorithm.

Input: A degree sequence $\mathbf{d}$ of length n.
Output: A graph $G(V, E)$ with nodes having degree sequence $\mathbf{d}$ or "No" if the input sequence is not realizable.

$V \leftarrow \{1, \ldots, n\}$, $E \leftarrow \emptyset$
if $\sum_i \mathbf{d}(i)$ is odd **then**
    Halt and return "No"
while 1 **do**
    if there exists $\mathbf{d}(i)$ such that $\mathbf{d}(i) < 0$ **then**
        Halt and return "No"
    if the sequence $\mathbf{d}$ are all zeros **then**
        Halt and return $G(V, E)$
    Pick a random node v with $\mathbf{d}(v) > 0$
    Set $\mathbf{d}(v) = 0$
    $V_{\mathbf{d}(v)} \leftarrow$ the $\mathbf{d}(v)$-highest entries in $\mathbf{d}$ (other than v)
    for each node $w \in V_{\mathbf{d}(v)}$ **do**
        $E \leftarrow E \cup (v, w)$
        $\mathbf{d}(w) \leftarrow \mathbf{d}(w) - 1$

Figure 1: ConstructGraph Algorithm

The output of the script is a csv file named `graph_friend_max_node_min_edge_max_edge.csv`.

Example
The following execution of the script:

```
python create_graph.py 1000 10 100 engwales_surname.csv
```

outputs a csv file named `graph_friend_1000_10_1000.csv`, where the first name of a row is a node and the remaining names are the links of this node.

Output

You are required to:

1. Implement the greedy algorithm explained above.

2. Test your implementation by executing the algorithm for several value of k on different graphs generated through the script.

3. For each test, assess whether it is possible to create a k-degree anonymous graph.

4. Analyze the relationship between the value of k, the size of the graph and the degree-anonymization cost (i.e., $D_A(\hat{\boldsymbol{d}}, \boldsymbol{d})$).

5. Visualize and analyze both the original and the anonymized graphs through the `networkx` tool [3] and grossly evaluate the loss of utility.

[3] `https://networkx.github.io`, the tutorial is available at: `https://networkx.github.io/documentation/networkx-1.10/tutorial/tutorial.html`

Contacts

Davide Caputo
Location (Valletta Puggia): Finsec Lab 320, 3rd floor, Via Dodecaneso 35, Genova.
Email: dave.caputo93@gmail.com

References

[1] K. Liu and E. Terzi, "Towards identity anonymization on graphs," *Proceedings of the 2008 ACM SIGMOD international conference on Management of data - SIGMOD '08*, p. 93, 2008. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1376616.1376629