Università di Genova
DIBRIS DIPARTIMENTO DI INFORMATICA, BIOINGEGNERIA, ROBOTICA E INGEGNERIA DEI SISTEMI

K Degree Anonymity

Degree anonymization by Dynamic Programming and construction of the Supergraph

Agnello Andrea, Rabellino Daniele, Viaggi Manuel

Graph Anonymization Problem - Introduction

Why removing the identities of the nodes doesn't always guarantee privacy?

Identity disclosure: the identity of the individual who is associated with the node is revealed.

Graph Anonymization Problem

1. First, starting from $\mathbf{d}$, we construct a new degree sequence $\widehat{\mathbf{d}}$ that is k-anonymous and such that the *degree-anonymization* cost

$$\mathrm{D_A}(\widehat{\mathbf{d}}, \mathbf{d}) = L_1(\widehat{\mathbf{d}} - \mathbf{d}),$$

is minimized.

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

$$L_1\left(\widehat{\mathbf{d}} - \mathbf{d}\right) = \sum_i \left|\widehat{\mathbf{d}}(i) - \mathbf{d}(i)\right|,$$

Degree Anonymization Problem

Given d, the degree sequence of graph G(V;E), and an integer k construct a k-anonymous sequence $d^{\wedge}$ such that $L(d^{\wedge}-d)$ is minimized.

Two possible solutions are:

Greedy Algorithm

- Solves the problem in time O(nk)
- Gives high-quality results

Dynamic Programming Algorithm

- Solves the problem in time O(n2) improvable to O(nk)
- Gives optimal results

Solution: Greedy Algorithm

$$C_{\text{merge}} = \left(\mathbf{d}(1) - \mathbf{d}(k+1)\right) + I\left(\mathbf{d}[k+2, 2k+1]\right),$$

and

$$C_{\text{new}} = I\left(\mathbf{d}[k+1, 2k]\right).$$

1. If **Cmerge > Cnew**, a new group starts with the **(k + 1)-th** node and the algorithm proceeds recursively for the sequence **d[k + 1; n]**.

2. Else 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.

Solution: Dynamic Programming

for $i < 2k$,

$$D_A(\mathbf{d}[1,i]) = I(\mathbf{d}[1,i]). \quad (2)$$

For $i \geq 2k$,

$$D_A(\mathbf{d}[1,i]) = \min \quad (3)$$
$$\left\{ \min_{k \leq t \leq i-k} \left\{ D_A(\mathbf{d}[1,t]) + I(\mathbf{d}[t+1,i]) \right\}, I(\mathbf{d}[1,i]) \right\}.$$

$$I(\mathbf{d}[i,j]) = \sum_{\ell=i}^{j} (\mathbf{d}(i) - \mathbf{d}(\ell)).$$

$D_A(\mathbf{d}[1,i])$

I(d[i,j]) be the degree anonymization cost when all nodes *i;i+1;. . .;j* are put in the same anonymized group.

Da (d[1; i]) be the degree-anonymization cost of subsequence d[1; i].

If i ≥ 2k, we check whether it is cheaper to split the sequence or not.

Dynamic Programming

To improve the running time of the DP algorithm from $O(n^2)$ to $O(nk)$ it is necessary to change the range in which to choose t and the result is this:

$$\mathrm{D_A}\left(\mathbf{d}[1,i]\right) = \min_{\max\{k,i-2k+1\}\leq t\leq i-k} \left\{\mathrm{D_A}\left(\mathbf{d}[1,t]\right) + I\left(\mathbf{d}[t+1,i]\right)\right\}. \tag{4}$$

Dynamic Programming Algorithm

```
Array degrees (Original): [97 97 96 89 83 78 76 76 75 74 74 72 72 67 65 65 65 65 64 64 64 64 64 63
 62 62 62 59 59 58 58 57 57 56 56 56 56 55 54 53 53 53 52 52 51 51 51 51
 50 49 49 48 48 47 46 45 45 44 43 43 43 42 42 42 41 41 40 38 38 38 38 38
 38 37 35 34 33 33 33 32 31 30 30 30 29 27 25 25 23 21 17 17 15 14 14 13
 12 11 10 10]

Array degrees (Dynamic Programming): [97, 97, 97, 97, 83, 83, 83, 83, 75, 75, 75, 75, 72, 72, 72, 72, 65, 65, 65, 65, 64
, 64, 64, 64, 62, 62, 62, 62, 59, 59, 59, 59, 57, 57, 57, 57, 56, 56, 56, 56, 53, 53, 53, 53, 51, 51, 51, 51, 50, 50, 50
, 50, 48, 48, 48, 48, 45, 45, 45, 45, 43, 43, 43, 43, 41, 41, 41, 41, 38, 38, 38, 38, 38, 37, 37, 37, 37, 33, 33, 33, 33
, 30, 30, 30, 30, 27, 27, 27, 27, 21, 21, 21, 21, 14, 14, 14, 14, 14, 14, 14]
```

The first output show the original degrees sequence.

The second output is the result of the application of the Dynamic Programming Algorithm on a graph of 100 nodes with K=4.

Graph Construction

DEFINITION 4. *Given input graph $G(V, E)$, we say that degree sequence $\hat{\mathbf{d}}$ is realizable subject to G, if and only if there exists a simple graph $\hat{G}(V, \hat{E})$ whose nodes have precisely the degrees suggested by $\hat{\mathbf{d}}$ and $E \subseteq \hat{E}$.*

By definition the new graph must have all the edges of the original graph, but this requirement may be too strict to satisfy. In many cases the new graph can contain most of the edges of the original graph, but not necessarily all of them.

Lemma 1: ConstructGraph

LEMMA 1. *([6]) A degree sequence* $\mathbf{d}$ *with* $\mathbf{d}(1) \geq \ldots \geq \mathbf{d}(n)$ *and* $\sum_i \mathbf{d}(i)$ *even, is realizable if and only if for every* $1 \leq \ell \leq n-1$ *it holds that*

$$\sum_{i=1}^{\ell} \mathbf{d}(i) \quad \leq \quad \ell(\ell-1) + \sum_{i=\ell+1}^{n} \min\{\ell, \mathbf{d}(i)\} \qquad (5)$$

With this rule we can realize the constructGraph, which satisfies part of the definition of the Graph Construction.

Lemma 2: SuperGraph

LEMMA 2. *Consider degree sequence* $\widehat{\mathbf{d}}$ *and graph* $G(V, E)$ *with degree sequence* $\mathbf{d}$. *Let vector* $\mathbf{a} = \widehat{\mathbf{d}} - \mathbf{d}$ *such that* $\sum_i \mathbf{a}(i)$ *is even. If* $\widehat{\mathbf{d}}$ *is realizable subject to graph* G *then*

$$\sum_{i \in V_\ell} \mathbf{a}(i) \leq \sum_{i \in V_\ell} \left(\ell - 1 - \mathbf{d}^\ell(i)\right) + \sum_{i \in V - V_\ell} \min\{\ell - \mathbf{d}^\ell(i), \mathbf{a}(i)\}, \quad (6)$$

To keep the edges of the original graph it is necessary create the Supergraph and if it exists this condition must be satisfied.

SuperGraph Algorithm

After the check of the inequality it proceeds iteratively and in each step it mantains the residual additional degrees a of the vertices. In each iteration it picks an arbitrary vertex v and adds edges from v to a(v) vertices of highest residual additional degree, ignoring nodes v' that already connected to v in G.

For every new edge (v, v'), a(v') is decreased by 1. If the algorithm terminates and output a graph, then this graph has degree sequence d^ and is a supergraph of the original graph.

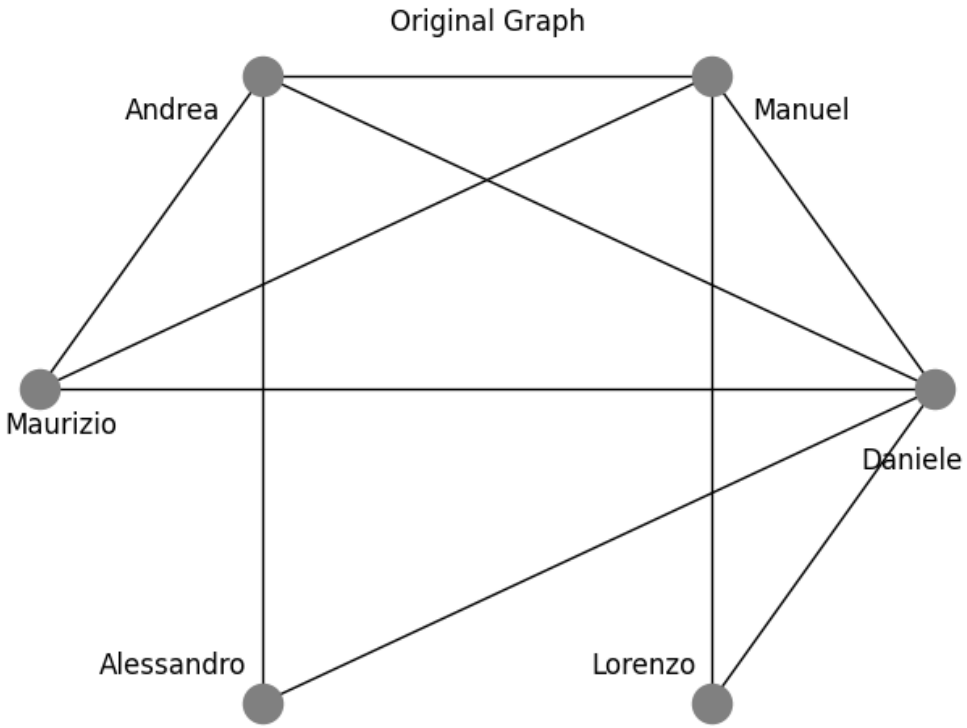
If the algorithm does not terminate, then it outputs "None", meaning that there might exist a graph, but the algoritm is unable to find it.

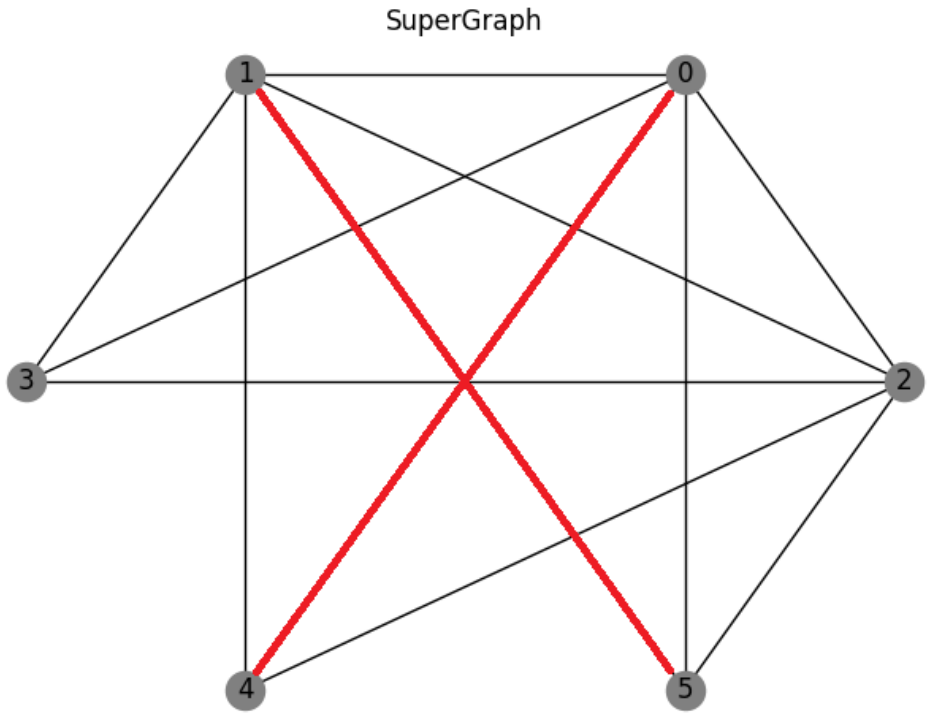
ConstructGraph and SuperGraph

- If the ConstructGraph returns null, it means that there is no graph for that degree anonymization.

- Instead if the SuperGraph algorithm does not return a graph, it does not necessarily mean that such a graph does not exists, but there is a scheme named probing scheme that forces the Supergraph to output the desidered k-degree anonymous graph with a little extra cost.

Original Graph and SuperGraph





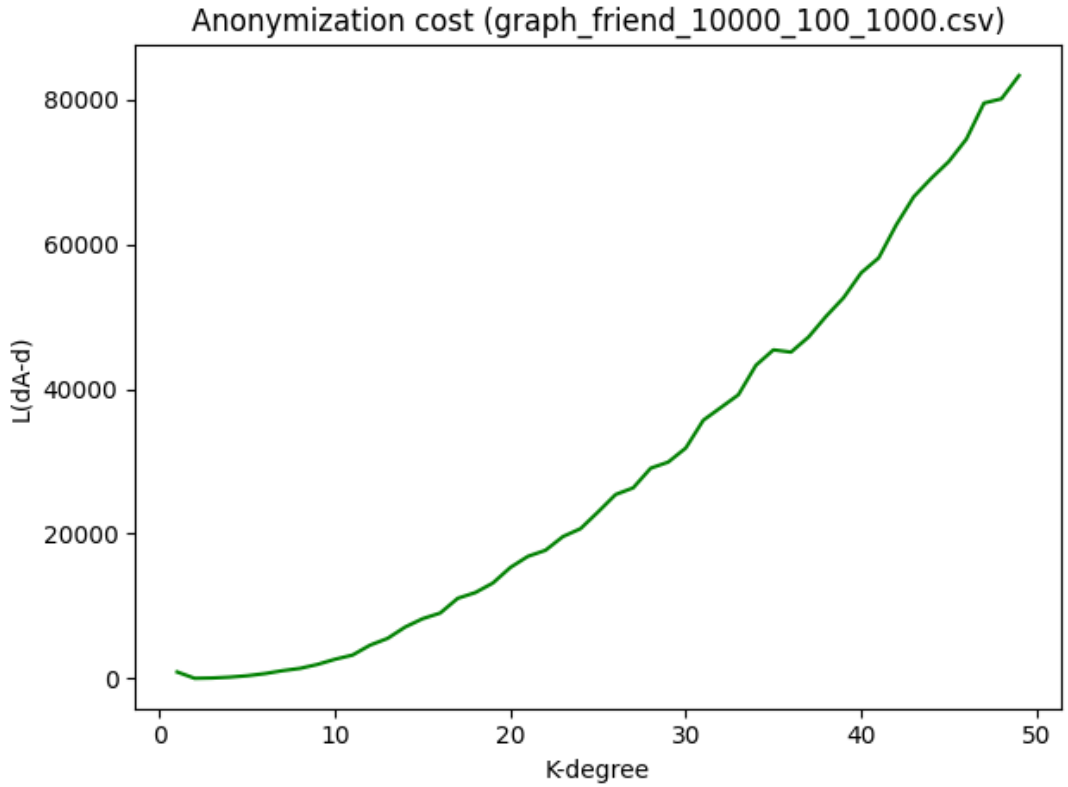
Anonymization cost

This image shows the increase of the value L(dA-d) with the increment of the value K.

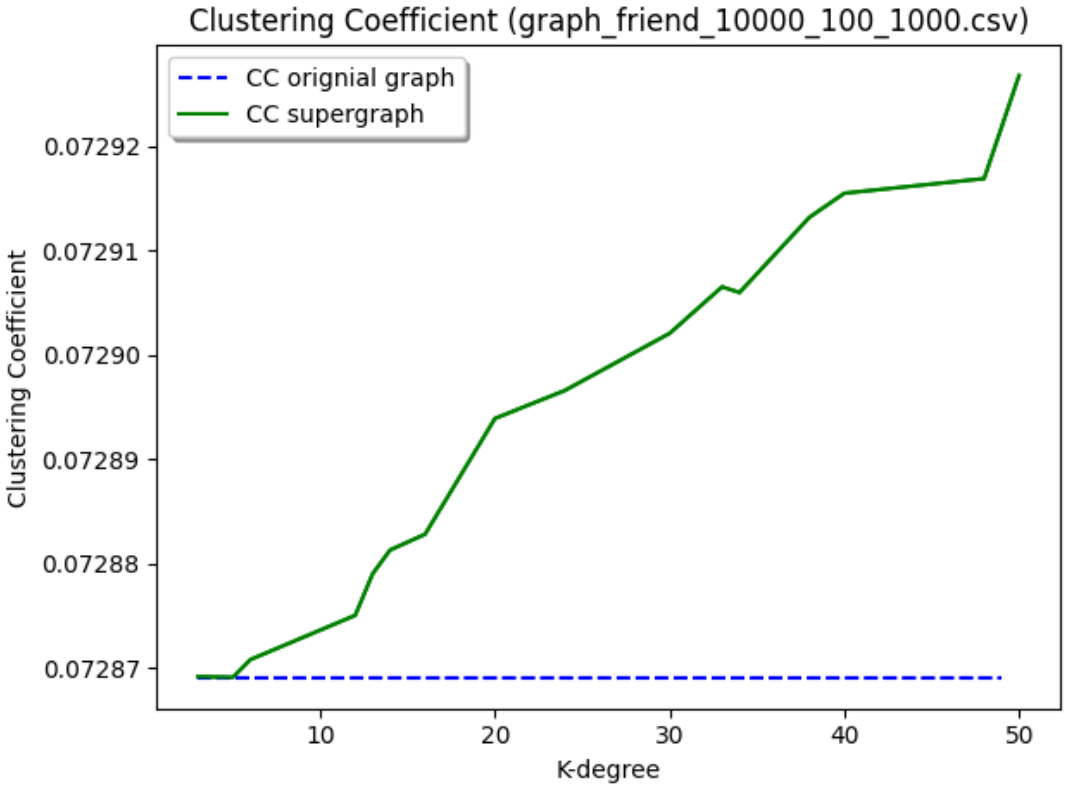
L is the norm of the vector of differences between the k-anonymous degree

sequence obtained by DP and the degree sequence of the original graph

The smaller the value of L, the better the qualitative performance of the algorithm.



Clustering Coefficient



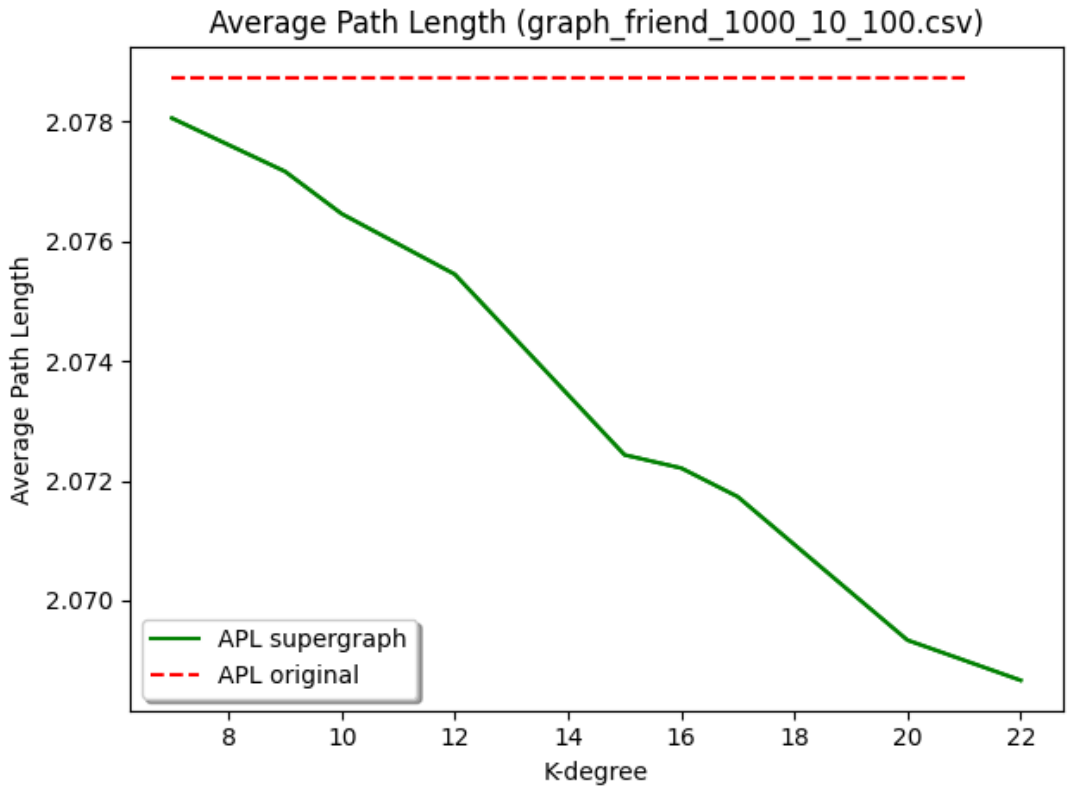
This image shows the CC of the original graph and the CC of the supergraph, we note that there is a small increase as k increases.

The trend seems to bee increasing, but the changes are small and negligible.

APL: Average path length

This image shows that the APL decrease with the increase of K since new connections are added.

We have made this test on the Graph with 1000 nodes.



Conclusions

We started with a very small graph created by us and then we tested our algorithms mainly on the laboratory datasets (graph_friend_1000_10_100 and graph_friend_10000_100_1000).

Without applying the probing scheme, the supergraph cannot always be obtained, but as seen in the last three images we have achieved quite satisfactory results.

Code

https://github.com/ManuManu97/k-degree