Data Protection and Privacy

Project Report

Towards Identity Anonymization on Graphs (Terzi and Lui on K-Degree anonymity)

Akshi Sharma(s.1001akshi@gmail.com) Jyoti Dhiman(idjyoti555i@gmail.com)

Introduction

This report is for the DPP project based on the paper by Terzi and Lui. In the project we have implemented the algorithms mentioned by the authors. We also took inspiration from the paper for the experiments done on the results of the algorithms. Our primary aim was to implement the algorithms as provided in the paper without using any libraries. Next, we evaluated the results and the performance of each algorithm with respect to some parameter. These were then compared to see how each algorithm works.

Topic

There is data everywhere. With today's crippling dependencies on technology the rate of increase in data is insurmountable. A lot of it can be useful for research purposes and therefore needs to be stored in specific databases depending on the kind of data. One such database is graphical database. Graph database and data structures are used everywhere to store data starting from social networks, VPN providers to a bank's customer information. While most of this data is used for analysis and research much of it can be misused to exploit data owners and stakeholders. To avoid any breach of privacy these databases need to be anonymized. It is a simple solution to a complex problem because if we simply anonymize all the data, it loses its usability for any-and-all analysis purposes. So now the problem is to anonymize data while maintaining the trade-off between privacy and utility.
To achieve this, we use the concept of "K-anonymity" which for graphs is translated to "K-degree Anonymity". The basic idea is that if in a dataset at least 'K' number of items have same information, or in case of graph data, have same degree it is K-degree anonymous.
So for a graph G(V, E), where "V" is the number of vertices/nodes and "E" is the number of edges, is k-degree anonymous if every node in V has the same degree as k-1 other nodes in V.

Figure 1 *The graph after anonymization is 2-degree anonymous as there are at least 2 nodes with same degree.*

Definitions

Degree Sequence

A degree of a vertex is the number of edges that originate or end on that node. We can collect all degrees corresponding to all the vertices and make a set. The set is called the degree sequence. It is sorted in the descending order with ***d[1] >= d[2]>=d[3]***… so on.

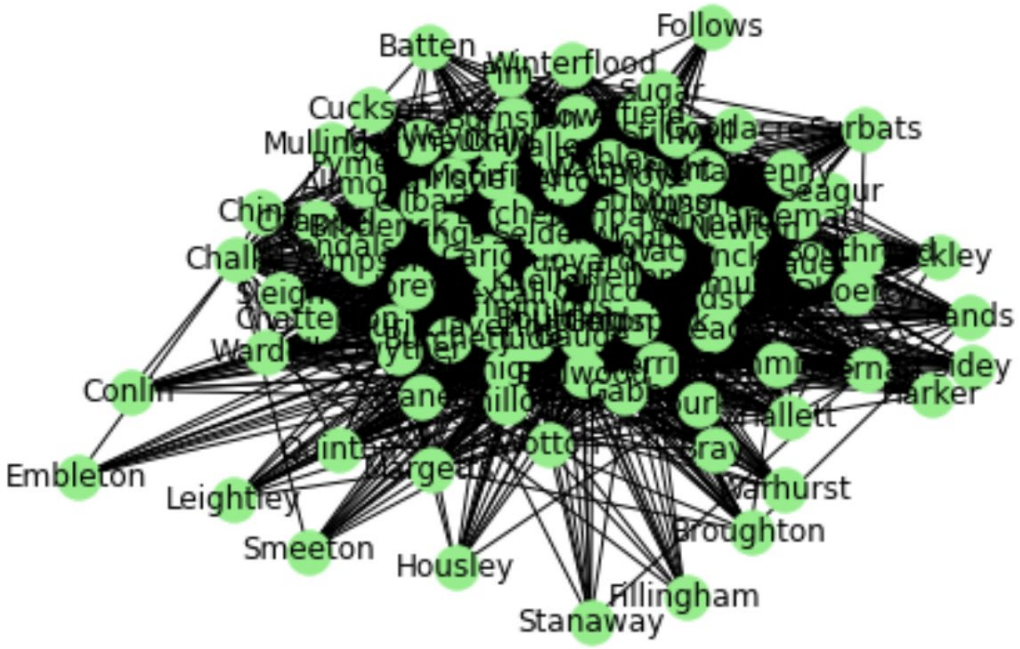
K-anonymous Sequence

A sequence of integers ‘d’ is k-anonymous if every distinct element in ‘d’ appears at least k times. A graph G (V, E) is k-degree anonymous if its degree sequence is k-anonymous.

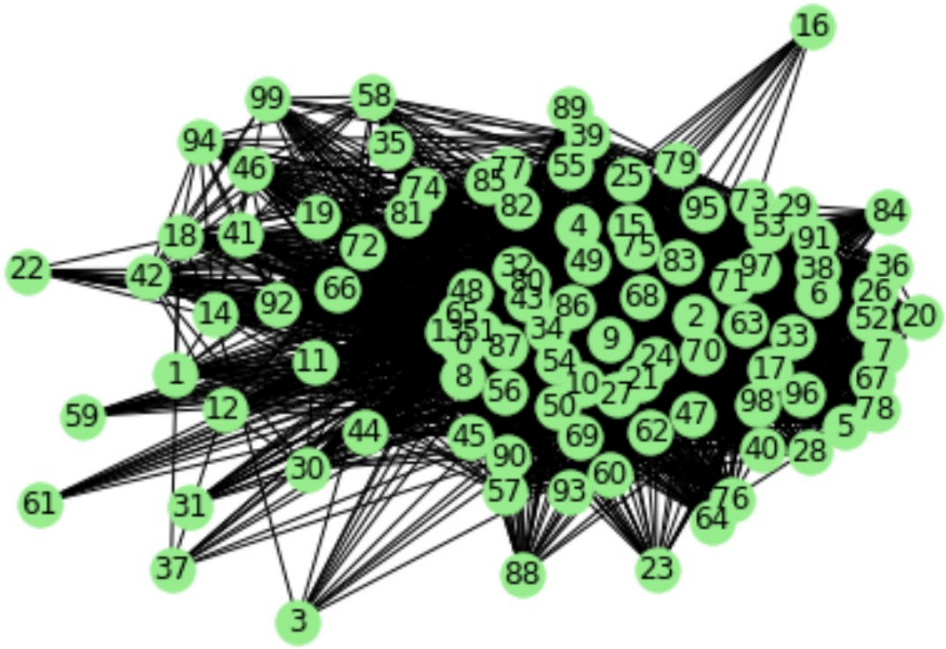
Cost function

To anonymize a graph the original degree sequence ‘d’ needs to be changed and turned into a k-anonymous sequence $\widehat{\mathrm{d}}$. There is cost to perform these changes which are calculated using the cost function called degree anonymization cost.

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



Original Graph



Anonymized Graph

K- degree Anonymization Algorithms

To implement k-degree anonymization following algorithms were used.

Greedy Algorithm

In the Greedy algorithm the priority is to achieve K anonymity quickly so it first forms a group consisting of the first k highest-degree nodes and assigns to all of them degree d[1] irrespective of the anonymization cost.
Then it checks whether it should merge the (k+1)-th node into the previously formed group or start a new group at position (k + 1). This decision is made by calculating the cost of merging (Cmerge) and the cost of creating a new group (Cnew). The one with less cost is chosen and the algorithm functions recursively to output a K degree sequence.

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

and

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

Dynamic programming

Unlike greedy algorithm Dynamic programming is implemented by breaking the problem into smaller recurring sub-problems. It computes the cost to turn a sub-sequence into k-degree anonymous sequence.
Da (d[1, i]) is the degree-anonymization cost of subsequence ***d[1, i]***.

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

I (d[i, j]) be the degree anonymization cost of putting all nodes ***i, i+ 1, . . . , j*** in the same group.

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

For ***i < 2k,*** it is impossible to construct two distinct anonymized groups each of size k. Since we only have one sequence, therefore both the costs ***Da (d[1, i])*** and ***I (d[i, j])*** are equal.

For ***i >= 2k,*** degree sequence can be broken down in smaller sub-sequences. So the costs ***Da (d[1, i])*** and ***I (d[i, j])*** are different and both combined give the total cost of creating sub sequences. The algorithm must decide between creating different groups or combining into one by choosing the action that costs less. Therefore, we choose the minimum of the two costs.

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

Optimised Dynamic programming

The dynamic programming algorithm requires a lot more computations. To improve the running time of the Optimised DP algorithm is suggested. The core idea is that if any group is larger than or equal to 2k, it can be broken into two subgroups with equal or lower overall degree-anonymization cost.
Since no anonymous group should be of size larger than ***2k – 1*** the values of ***I (d[i, j])***, does not have to consider all the combinations of ***(i, j)*** pairs, but only values such that

k ≤ j − i + 1 ≤ 2k – 1, for j
t's in the range max{k, i − 2k + 1} ≤ t ≤ i – k, for i

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

Datasets

We have used the Friends dataset (100_10_100 and 1000_10_100). The other files of this dataset were too large for the system to run. We also used real world datasets like football,

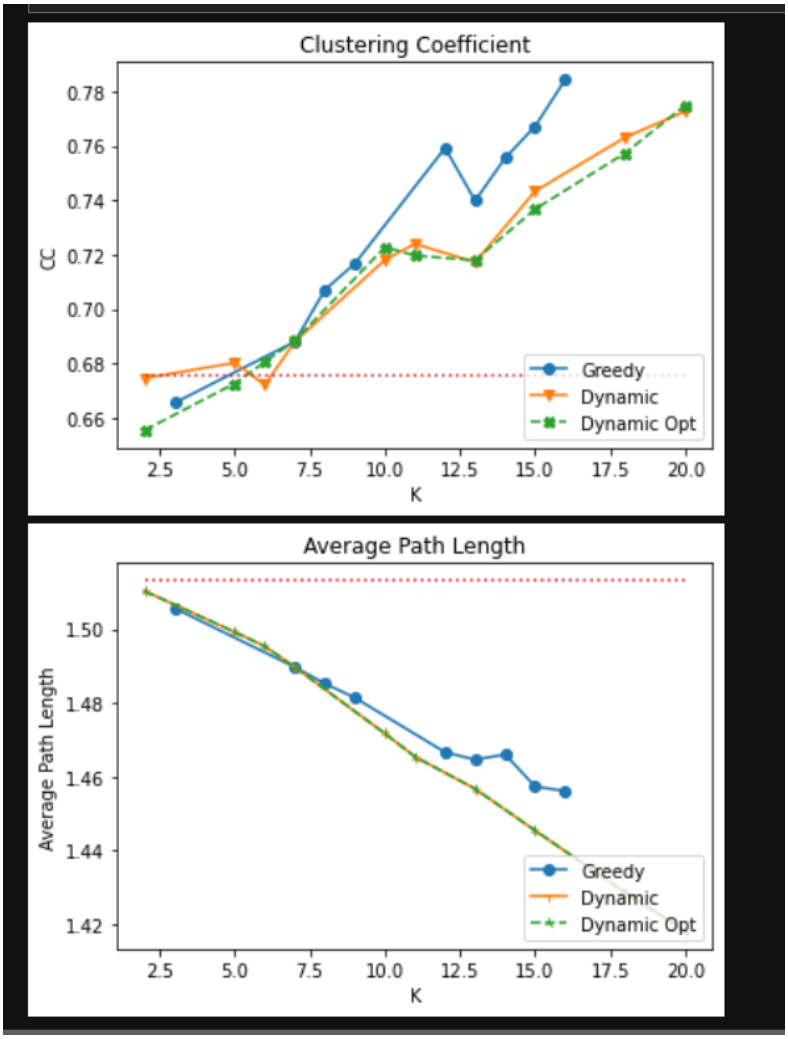
dolphins and polbooks. For experiments we primarily focused on Friends, Les Misérables real dataset and Albert Barabasi synthetic dataset.

Experiments and Results

For experiments we took inspiration from the authors and computed clustering coefficients and Average path length as a function of K for anonymized graphs.

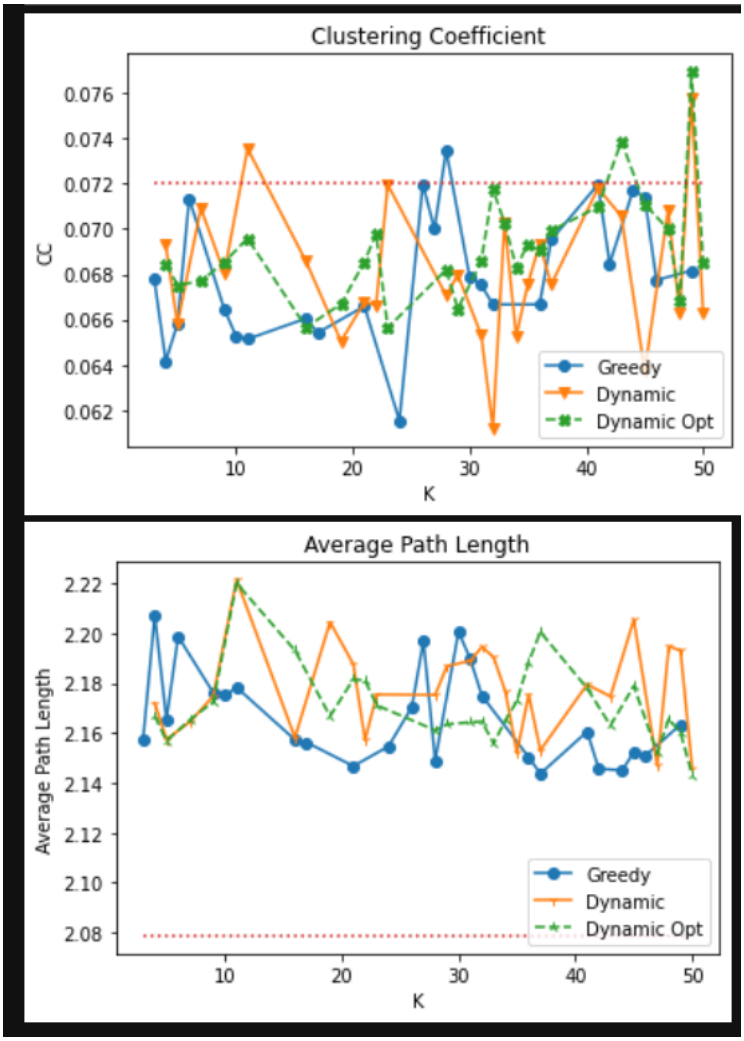
Clustering coefficient – The reasoning was that more clustering would lead to more anonymization. It was observed in our tests for small values of nodes (100). However, no specific pattern was observed on increasing the nodes to 1000

Average path length- The reasoning was that addition of edges will lead to smaller shortest paths. As expected with increasing K value the path length got smaller but increase in number of nodes did not show anything clearly.

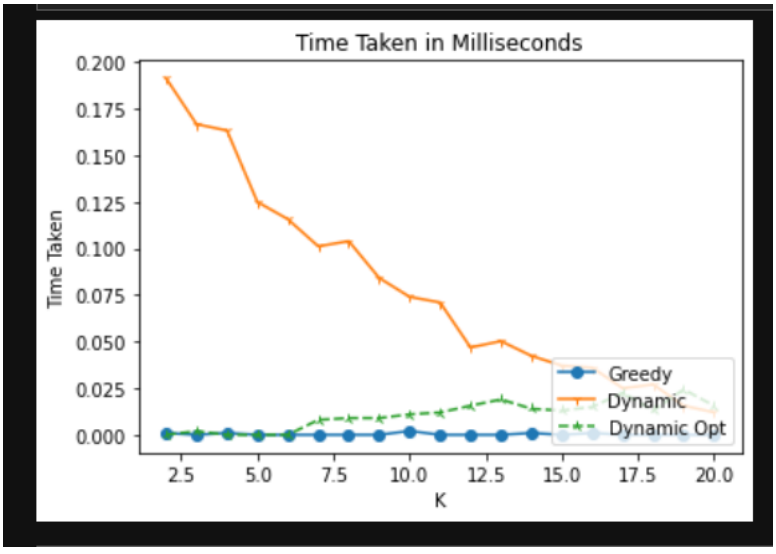


For 100 nodes

For 1000 nodes

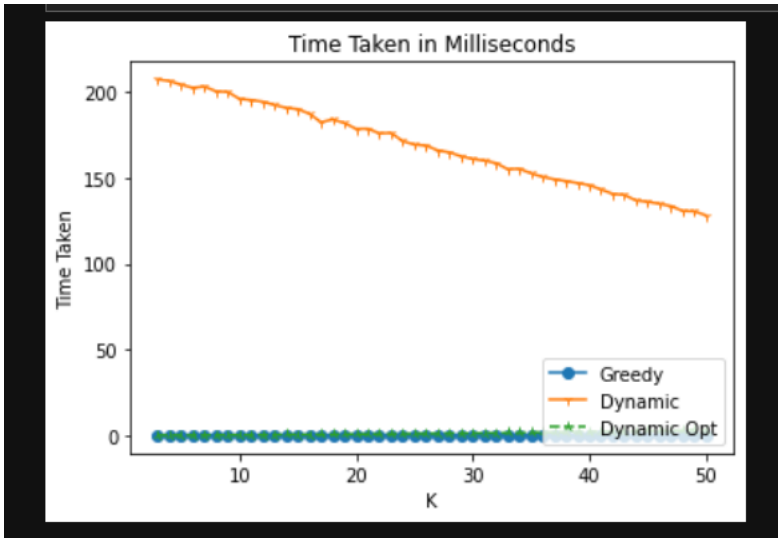


Time complexity – We tested the time complexity for all the algorithms w.r.t. value of K. As expected, greedy was the fastest with Dynamic Optimised close behind and nearly the same with increase in nodes. Dynamic programming got better with increase in K value but the decrease in time complexity got slower in higher number of nodes.



For 100 nodes

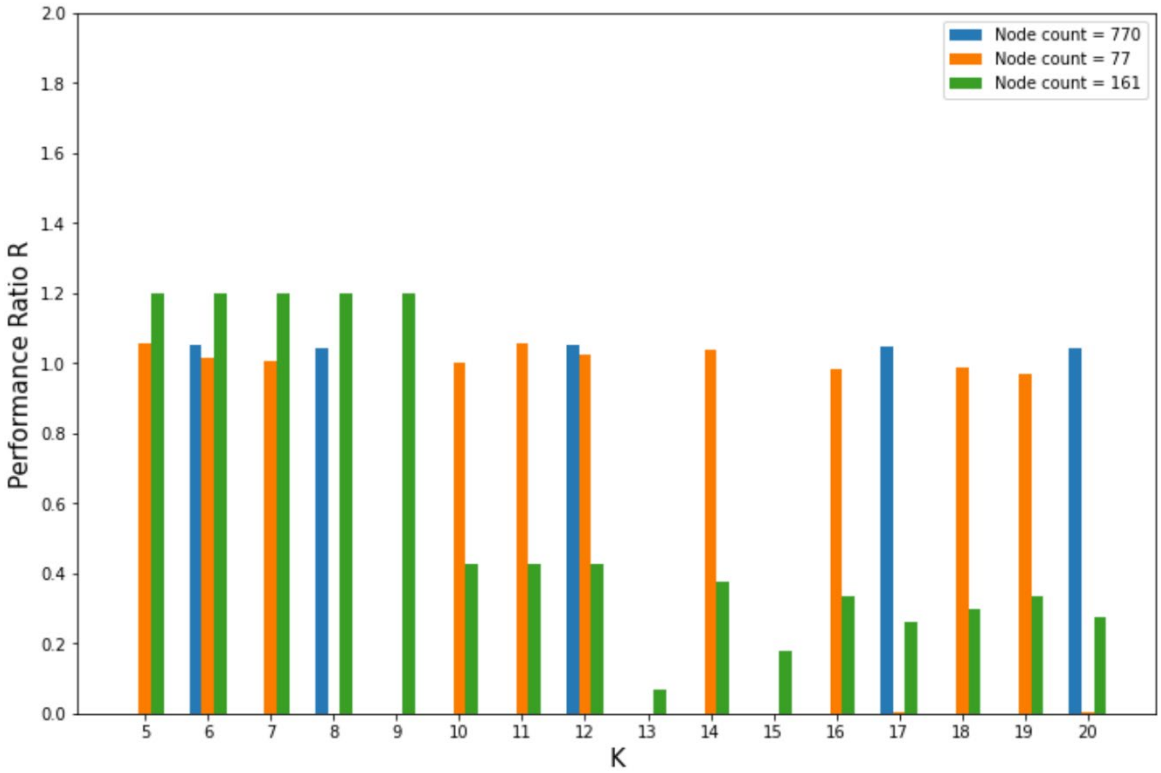
For 1000 nodes



Performance ratio R – The performance ratio R is the ratio of Cost of changing a degree sequence to anonymised degree sequence.

$$R = \frac{L_1(\hat{\mathbf{d}}_{\text{greedy}} - \mathbf{d})}{L_1(\hat{\mathbf{d}}_{\text{dp}} - \mathbf{d})}, \text{ where } \mathbf{d} \text{ is the input degree sequence;}$$

We calculated R of greedy algorithm wrt to Optimised DP as it performs better than DP. We deliberately chose the number of nodes of synthetic data to be 10 times of real data for comparison. The closer R is to 1, the better the performance of the Greedy algorithm.



Albert-Barabasi-770; Les miserables-77, Friends-161