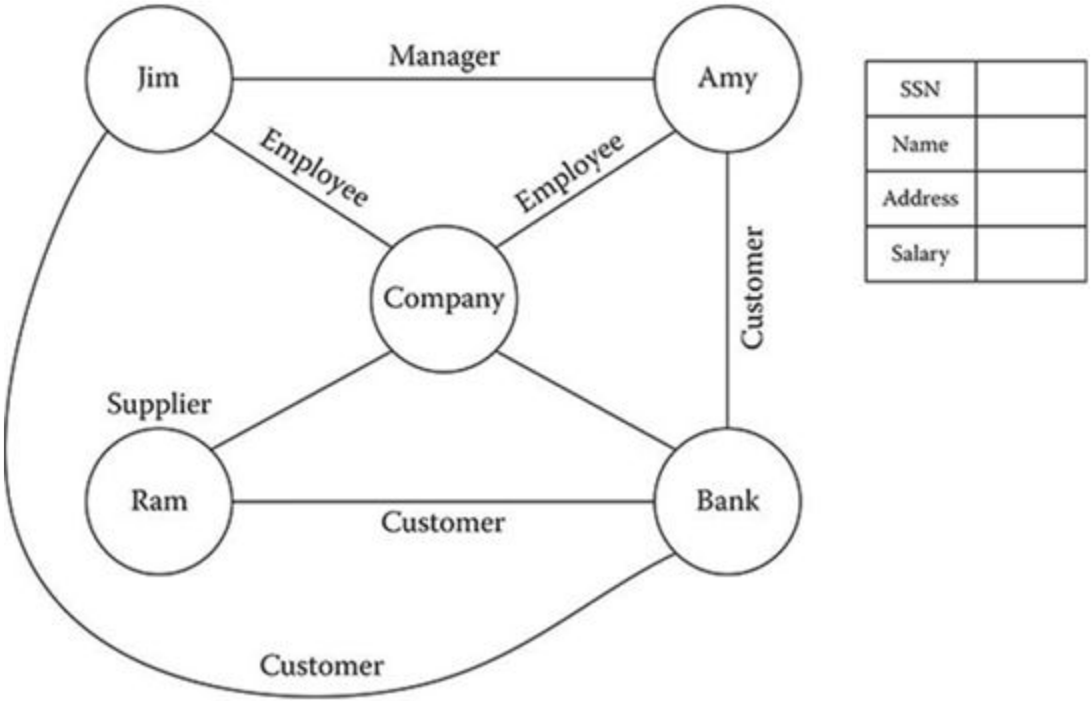
Anonymizing Graphs: k-Degree Anonymity

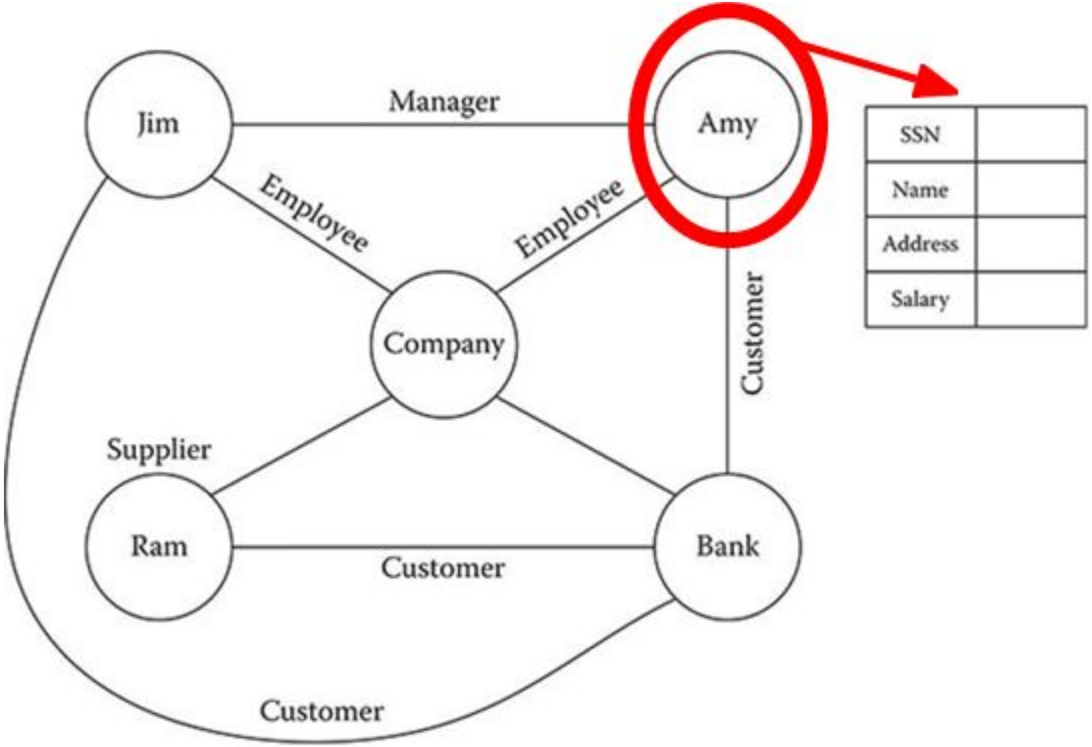
UNIVERSITÀ DEGLI STUDI DI GENOVA | Dibris

Francesco Pagano
s4192013@studenti.unige.it

Graphs


Jim
Manager
Amy
Employee
Employee
Company
Customer
Supplier
Ram
Bank
Customer
Customer
SSN
Name
Address
Salary

Graphs



Jim
Manager
Amy
Employee
Employee
Company
Customer
Supplier
Ram
Customer
Bank
Customer
SSN
Name
Address
Salary

Graphs vs. Multidimensional Data

- In multidimensional data each record or tuple can be transformed independent of each other.

- In graph data any change in the nodes or edges affects the characteristics of the graph and also the utility of the anonymized graph.
    - Vertex properties, Vertex Labels, Link relationship
    - Graph metrics

How Protect?

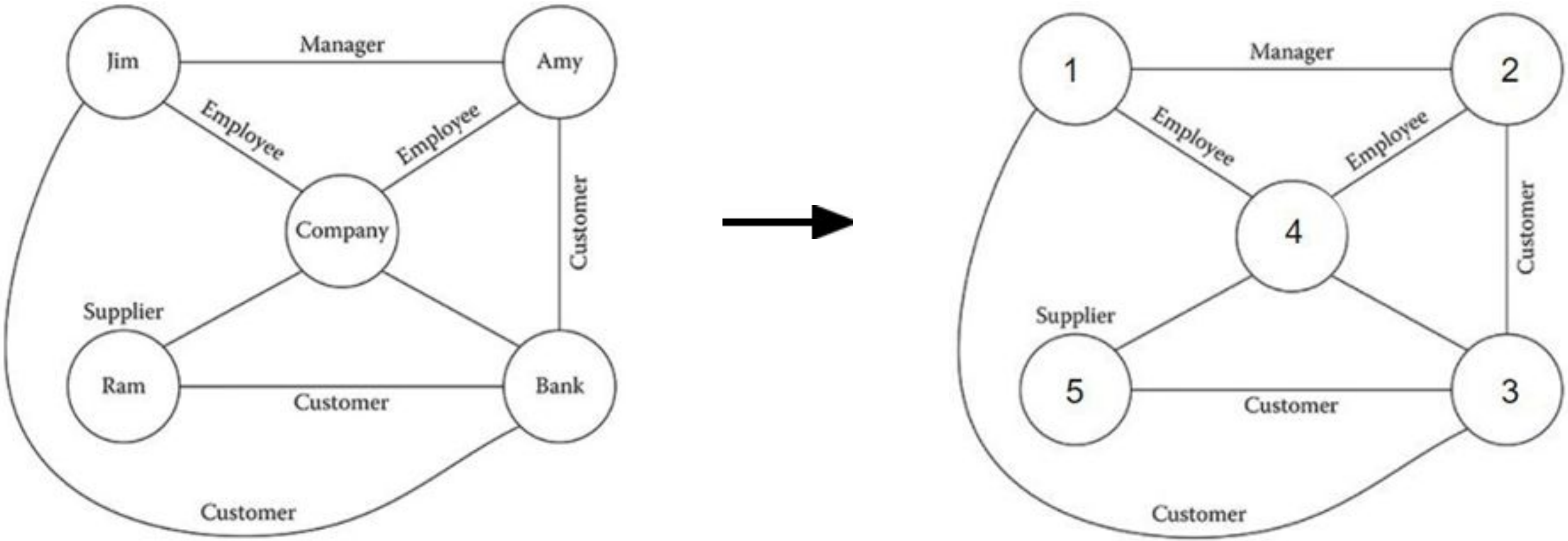
- Identity Protection → entities identification (EI in multidimensional data)
- Content Protection → entities information (QI, SD in multidimensional data)
- Link Protection → Relationship between entities

How to Anonymize?

- Naive anonymization → effective when adversary has no background knowledge
- Random Perturbation or graph modification → utility loss
- Clustering → grouping similar objects (i.e., vertices, edges, vertices-edges)

Identity Protection: Naive Anonymization

- Identiers (EI) are replaced by random values → high utility, low privacy. Weak against external knowledge.



Identity Protection: Graph Modification

- Idea: the degree of a node is informative (i.e., social networks) → it could allow to reveal identities

- **k-degree Anonymity** for each node v there exists other k - 1 nodes with the same degree as v.

k-Degree: Problem

- **Problem Formulation**
  - **G(V,E):** *simple graph*
  - $\mathbf{d_g}$: *denotes the degree sequences of G* ($|d_g| == n == |V|$)
  - $\mathbf{d_g(i)}$: *is the degree of the i-th node of G*

- **Assumption**
  - Entries in d are ordered in decreasing order of the degrees $d_g(1) >= d_g(2) >= \ldots >= d_g(n)$
  - d[i; j] (for i < j) denotes the subsequence of dg that contains elements i ; i + 1; ... ;j - 1; j

- **Definition**
  - A vector of integers v is k-anonymous, if every distinct value in v appears at least k times.
    - For example, vector v = [5, 5, 3, 3, 2, 2, 2] is 2-anonymous
  - A graph G(V; E) is k-degree anonymous if the degree sequence of G, dg ,is k-anonymous.

Example of k-degree graph

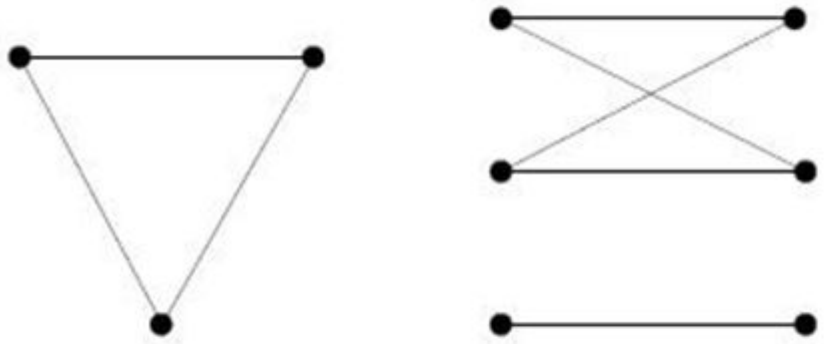


Figure 1: Examples of a 3-degree anonymous graph (left) and a 2-degree anonymous graph (right).

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

$$\text{DA}(\widehat{\mathbf{d}}, \mathbf{d}) = L_1(\widehat{\mathbf{d}} - \mathbf{d}),$$

   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^ such that L1(d^ - d) is minimized.

- **Solutions**
  - Dynamic Programming Algorithm
  - Greedy Algorithm

Dynamic Programming Algorithm

for $i < 2k$,

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

For $i \geq 2k$,

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

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

Dynamic Programming Algorithm (1)

for $i < 2k$,

$$\textsc{Da}\left(\mathbf{d}[1,i]\right) \;=\; I\left(\mathbf{d}[1,i]\right). \qquad (2)$$

- Equation (2). When i < 2k, it is impossible to construct two different anonymized groups each of size k. As a result, the optimal degree consists of a single group in which all nodes are assigned the same degree equal to d(1)

Dynamic Programming Algorithm (2)

For $i \geq 2k$,

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

- Equation (3) handles the case where i >= 2k. In this case, the degree-anonymization cost for the subsequence d[1; i] consists of optimal degree-anonymization cost of the sub-sequence d[1; t], plus the anonymization cost incurred by putting all nodes t + 1;...; i in the same group (provided that this group is of size k or larger).

Greedy Algorithm (1)

1. The Greedy algorithm first forms a group consisting of the first k highest-degree nodes and assigns to all of them degree d(1).

2. 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)

Greedy Algorithm (2)

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

and

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

1. If $C_{merge} > C_{new}$, 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.

Construct Graph Algorithm

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**
    $E \leftarrow E \cup (v, w)$
    $\mathbf{d}(w) \leftarrow \mathbf{d}(w) - 1$

Complete the algorithm

Download the code from Aulaweb and complete it

```python
def dp_graph_anonymization():
    # complete this function
    do_stuff()
```

```python
def greedy_rec_algorithm():
    # complete this function
    do_stuff()
```