Preserving Privacy in Social Networks Against Neighborhood Attacks

Bin Zhou and Jian Pei

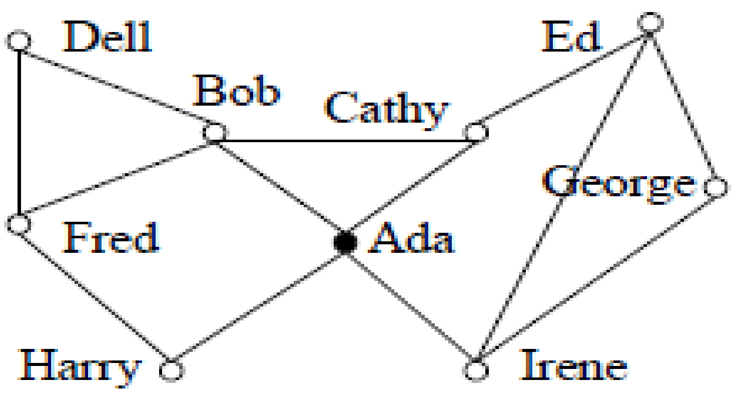
Presentation By
Naveed Alam

Social Networks

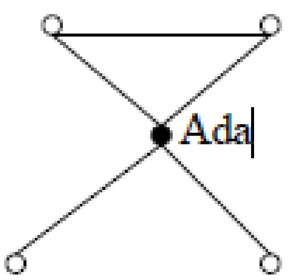
- It is inherent in people to socialize;
- Social networks are groups of nodes and links; nodes – actors, links – dependencies
- Increased interest in social networks in recent years
- Social network data is published and made available – Eg: “How to search a social network”, “Group formation in large social networks: membership, growth, and evolution”

Neighborhood Attack

- Removing node and edge labels does not protect privacy
- Having information about neighbors of a target victim and the relationship among the neighbors, it is possible to re-identify the target victim in an anonymized network
- Using neighborhood attack, can analyze the connectivity of the target node and its relative position in the network

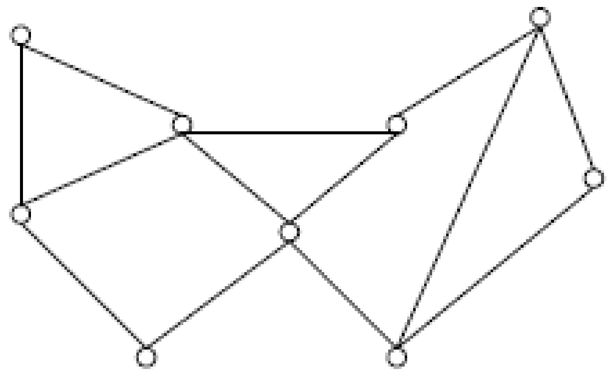


(a) the social network

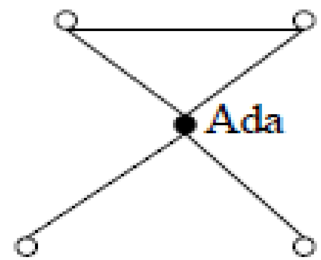


(c) the 1−neighborhood graph of Ada

Neighborhood Attack

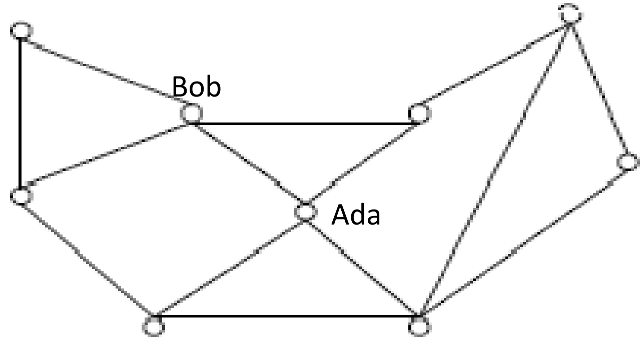


(b) the network with anonymous nodes



(c) the 1−neighborhood graph of Ada

Privacy can be provided by using the k – anonymity model



(d) privacy−preserved anonymous network

Challenges

- It is more difficult to anonymize social network data than relational data

- Measuring information loss in anonymizing social network data is difficult

- Anonymizing social networks is challenging. Removing/adding nodes and edges changes the properties of the network

Problem Definition

- $G = (V,E,L,\mathcal{L})$

$V \longrightarrow$ set of vertices

$E \longrightarrow$ set of edges ; $L \longrightarrow$ set of labels

$\mathcal{L}$ -> labeling function; assigns $V \longrightarrow L$

For graph G, V(G) = set of vertices

E(G) = set of edges

L_G = set of labels

$\mathcal{L}$ = *labeling function in G*

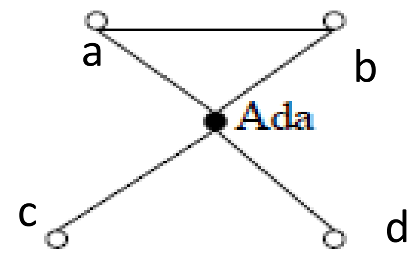
- Items in the label set L form a hierarchy
- Labels can be specific descriptions or generalized terms
- * is the most general category generalizing all labels
- If one label is more general than the other, it can be written as $l_1 \prec l_2$

  E.g.: l_1 = doctor and l_2 = optometrist then

  $l_1 \prec l_2$

1-Neighborhood

- Neighborhood of u in G is represented as
  $Neighbor_G(u) = G(N_u)$ where $N_u = \{ v | (u,v)$ is an edge in $G\}$



(c) the 1−neighborhood graph of Ada

- $G=(V,E,L,\mathcal{L})$ is a social network; $H=(V_H,E_H,L\,\mathcal{L})$; instance of H in G is $(H^{|}, f)$ where $H^{|}=(V,E,L,\mathcal{L})$ is a subgraph in G such that $f{:}V(H) \rightarrow V(H^{|})$ is a bijection
- Labels in $H^{|}$ can be more general than in H

Problem Formulation

- To anonymize a graph G, no new nodes are created thus preserving the global structure
- Adversary is assumed to have background knowledge i.e. information about the neighborhoods of some nodes
- If $Neighbor_G$ (u) has k instances in G' , G' is an anonymization of G, then u can be re-identified in G' with confidence 1/k

K-anonymity

Theorem 1 (k-anonymity): Let G be a social network and $G^{|}$ an anonymization of G. If $G^{|}$ is k-anonymous, then with the neighborhood background knowledge, any vertex in G cannot be re-identified in $G^{|}$ with confidence larger than 1/k.

$G^{|}$ does not contain fake vertices

All edges in G are also in $G^{|}$

Neighborhood Extraction and Coding

- If u ε V(G), subgraph C is a neighborhood component of u if C is a maximal connected subgraph

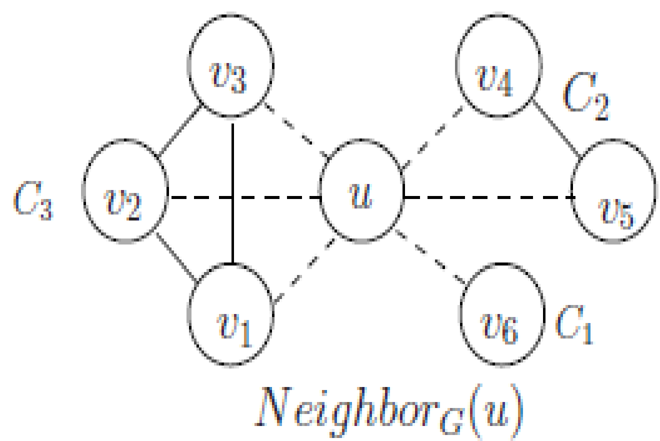
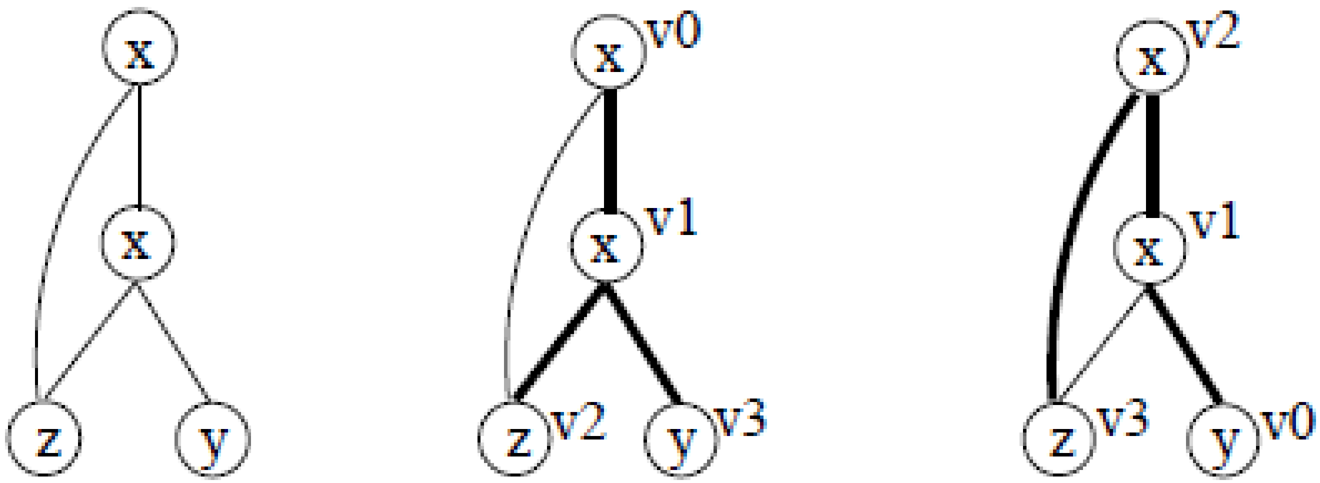


Fig. 2. Neighborhood and neighborhood components (the dashed edges are just for illustration and are not in the neighborhood subgraph).

Neighborhood Extraction and Coding

- DFS-tree can be used to code the vertices and edges in a graph



(a) Graph G (b) DFS-tree T_1 (c) DFS-tree T_2

Fig. 3. DFS codes, starting from different vertices.

Neighborhood Extraction and Coding

- A linear order $\prec$ on the edges in G can be defined given two edges

  $e = (v_i, v_j)$ and $e^| = (v_k, v_l)$ as

1. e and $e^|$ are forward edges ($j<l$)
2. e and $e^|$ are backward edges ($i<k$)
3. When e is a forward edge and $e^|$ is a backward edge ($j<=k$)
4. When e is a backward edge and $e^|$ is a forward edge ($i<=l$)

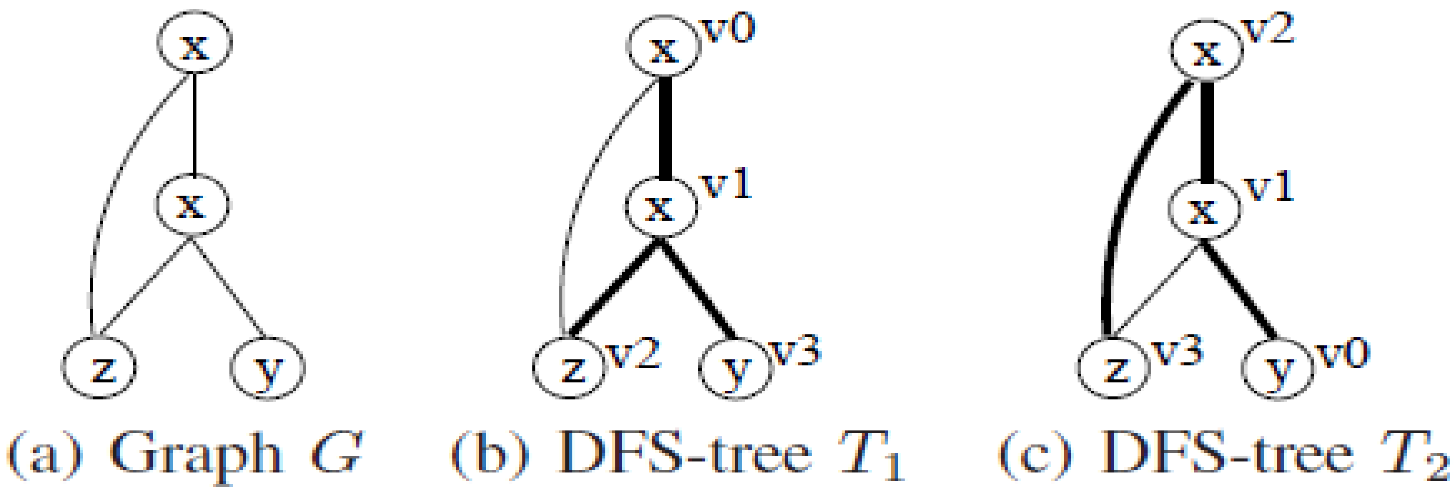


Fig. 3. DFS codes, starting from different vertices.

- DFS *code(G; T1) = {(v0; v1; x; x)-(v1; v2; x; z)-(v2; v0; z; x)-(v1; v3; x; y)}*
- *DFS code(G; T2) = {(v0; v1; y; x)-(v1; v2; x; x)-(v2; v3; x; z)-(v3; v1; z; x)}*
- *code(G; T1) < code(G; T2)*
- *Minimal DFS(G) = code(G; T1)*

Neighborhood Extraction and Coding

- Neighborhood Component Code

In a social network *G, for vertex u ε V (G), the neighborhood component code of Neighbor G(u) is a vector*

NCC(u) = (DFS(C1);......;DFS(Cm)) where *C1;.......;Cm* are the neighborhood components of *NeighborG(u)*

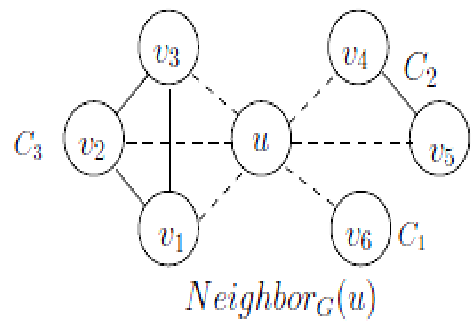


Fig. 2. Neighborhood and neighborhood components (the dashed edges are just for illustration and are not in the neighborhood subgraph).

- *NCC(u) = (DFS(C1);DFS(C2);DFS(C3)).*
- *Theorem (Neighborhood component code): For two vertices u; v ε V (G) where G is a social network, NeighborG(u)* and *NeighborG(v) are isomorphic if and only if NCC(u) = NCC(v).*

Anonymization Quality Measure

- Consider a vertex *u of label l1, where l1 is at the leaf level* of the label hierarchy, i.e., *l1 does not have any descendant.*
- Normalized Certainty Penalty-

  Suppose *l1 is generalized to l2 for u where l2* $\prec$ *l1*
  Let size(l2) be the number of descendants of l2 that are leafs in the label hierarchy, and *size(*) be the total number of leafs* in the label hierarchy.
  Then, the *normalized certainty penalty* of *l2 is NCP(l2) = size(l2)/size(*) .*

Anonymization Cost

- Consider two vertices *u1, u2 ε V (G) where G is a social* network
- Suppose *NeighborG(u1) and NeighborG(u2) are* generalized to *NeighborG0 (A(u1)) and NeighborG0 (A(u2))* such that *NeighborG0 (A(u1)) and NeighborG0 (A(u2)) are* isomorphic.
- Let *H = NeighborG(u1) U NeighborG(u2) a*nd

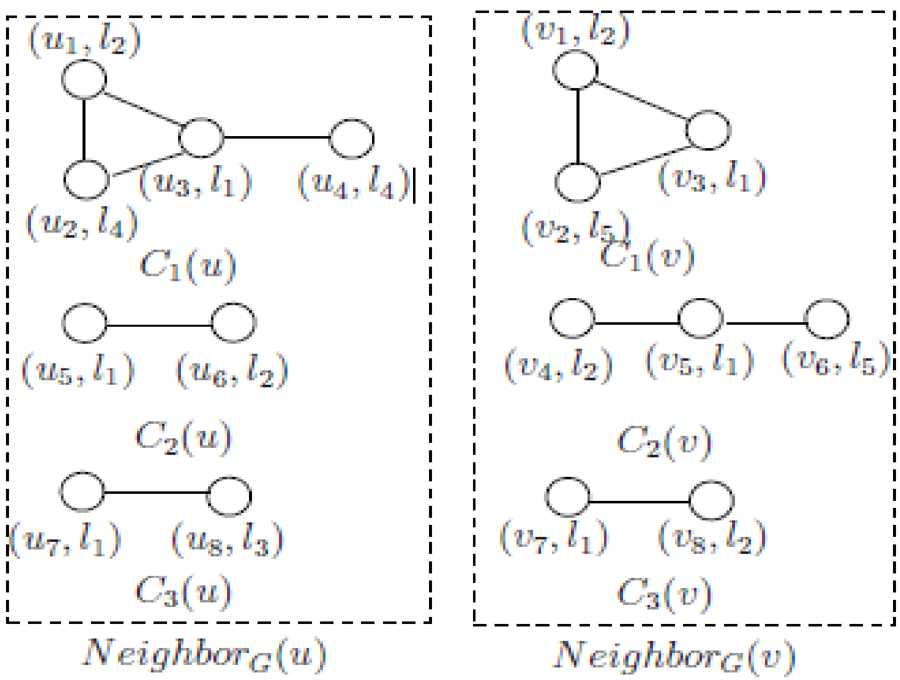
  H0 = NeighborG0 (A(u1)) U NeighborG0 (A(u2)). The anonymization cost is

  α(NCP)+β(information loss due to adding edges)+΄Υ(number of vertices linked to anonymization neighborhood to achieve k-anonymity)

  The parametrs are weights specified by users.

Anonymizing Neighborhoods

- Two neighborhood components match each other if they have the same minimum DFS code and are marked as "matched"



C2(u) = C3(v)

Anonymizing Neighborhoods

- If two components do not match then similarity is found between the components by comparing the similar (vertices, label) pairs.
- If multiple matching vertex pairs, choose the one with highest degree
- If no pairs can be found then matching requirement is relaxed until a match is found
- The vertex with the minimum anonymization cost is chosen and a breadth-first search is performed to match all vertices
- Similarity between two components is based on anonymization cost

Anonymizing Neighborhoods

When a vertex needs to be introduced then

1. First consider unanonymized vertices in G
2. Vertex with smallest degree has highest priority
3. If more than one vertex have smallest degree choose the one with lowest anonymization cost
4. If unanonymized vertex cannot be found, select an anonymized vertex satisfying above requirements

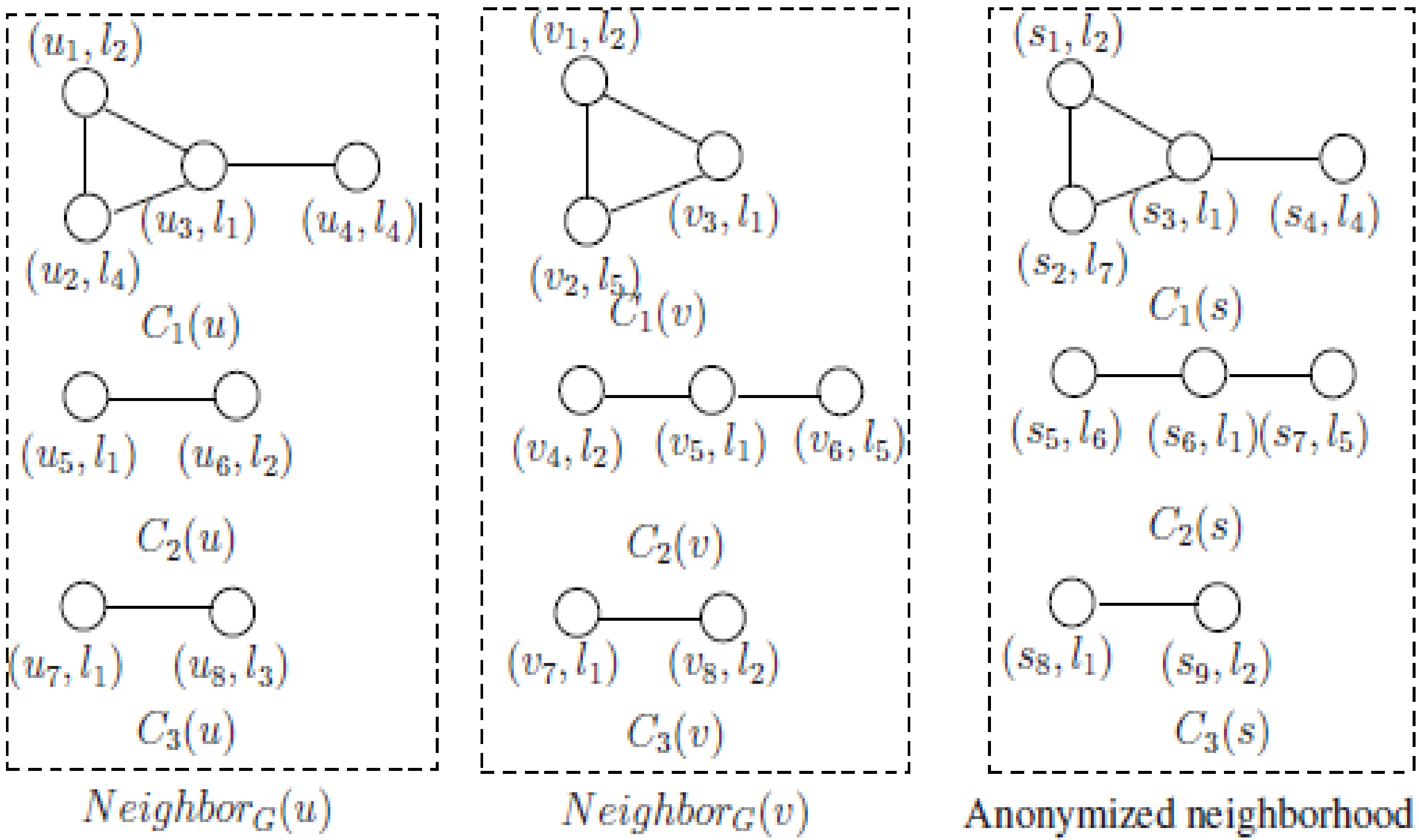


Fig. 4. Anonymizing two neighborhoods.

Anonymizing a Social Network

- Input: a social network *G = (V;E), the anonymization requirement* parameter *k, the cost function parameters*

- Method:
  1: initialize *G0 = G;*
  2: mark *vi ε V (G) as "unanonymized";*
  3: sort *vi ε V (G) as VertexList in neighborhood size descending* order;
  4: WHILE (*VertexList != 0) DO*
  5: let *SeedVertex = VertexList.head() and remove it* from *VertexList;*

Anonymizing a Social Network

6: FOR each *vi ε VertexList DO*

7: calculate *Cost(SeedVertex, vi) using the anonymization* method for two vertices;

END FOR

8: IF (*VertexList.size() < 2k - 1) DO*

Let *CandidateSet contain the top k - 1 vertices with the* smallest *Cost;*

9: ELSE

10: let *CandidateSet contain the remaining unanonymized* vertices;

Anonymizing a Social Network

11: suppose *CandidateSet= (U1,…….. Um), anonymize Neighbor(SeedVertex) and Neighbor(u1)*

12: FOR *j = 2 to m DO*

13: anonymize *Neighbor(uj) and {Neighbor(SeedVertex), Neighbor(u1),……,Neighbor(uj¡1)}*

mark them as “anonymized”;

14: update *VertexList;*

END FOR

END WHILE

Empirical Evaluation

- Co-authorship dataset from KDD Cup 2003 containing 57,448 nodes and 120,640 edges.
- Anonymization by removing labels and generalizing labels
- As k increases, no. of vertices violating k-an

k	Removing labels	Generalizing to affiliations
5	1.3%	12.7%
10	3.9%	16.1%
15	7.1%	19.4%
20	12.0%	23.2%

TABLE I

THE PERCENTAGES OF VERTICES VIOLATING k-ANONYMITY IN THE CO-AUTHORSHIP DATA.

Anonymization Performance

- Synthetic datasets were generated with average vertex degree 3 to 8 and no. of vertices varying from 25000 to 30000
- Keeping *β* equal to 1 and varying *α, ΄Ƴ shows that adding less edges is more desirable in anonymizing a social network*
- *When α =100, ΄Ƴ =1.1 the number of edges added is small and NCP is moderate*

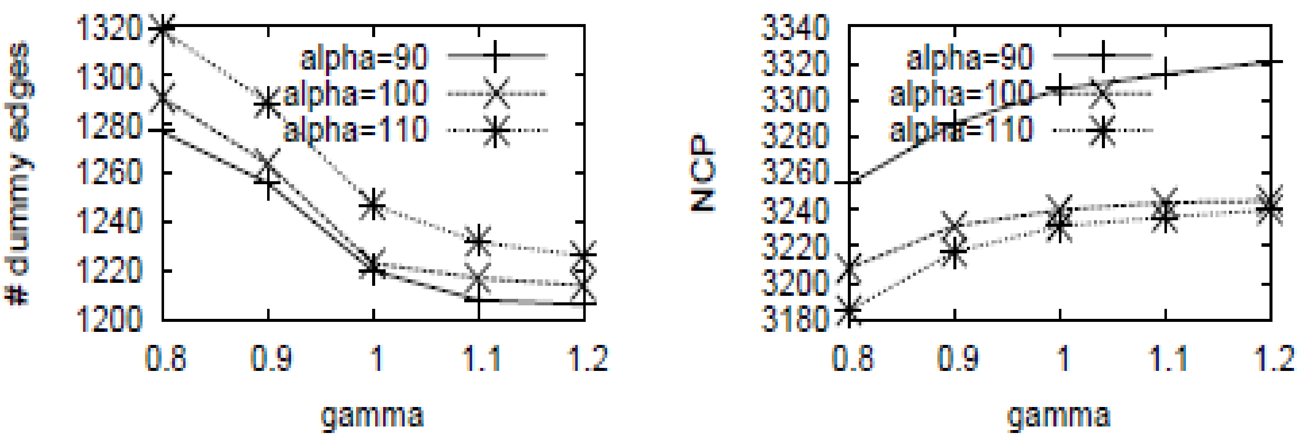
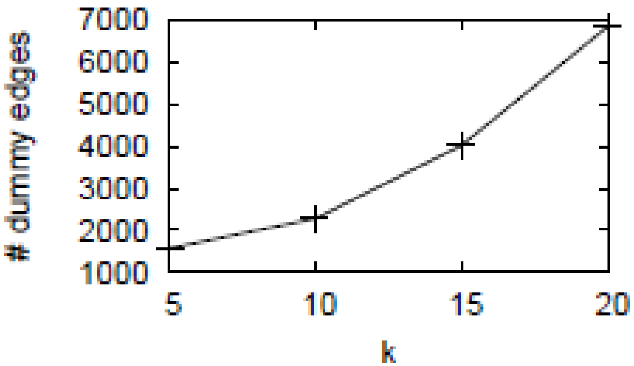
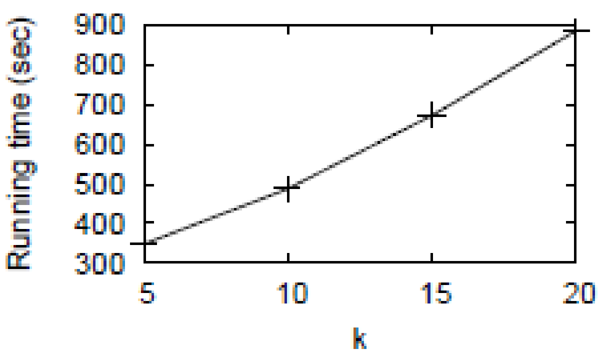


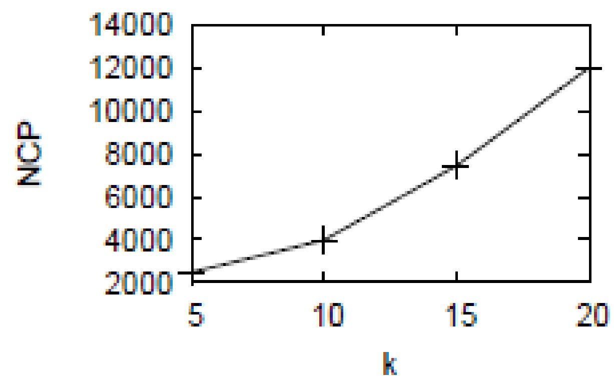
Fig. 6. The effect of parameters in anonymization quality measure.

Anonymization of KDD Dataset

- Three label hierarchy level was used.
- The total number of edges added is less than 6% of the original number of edges upto k=20







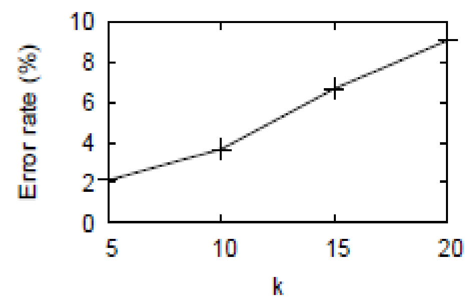


Fig. 10. Query answering on the KDD Cup 2003 co-authorship data set.

Conclusions

- The k-anonymity model can be used to provide anonymity to social network data by anonymizing 1-neighborhood of each vertex
- An adversary can indentify a victim in a group of anonymized vertices all of which share some sensitive information
- Future research can be to introduce l-diversity and anonymization of d-neighborhoods