Privacy in Social Networks: Structural identity disclosure

1

Methods based on k-anonymity

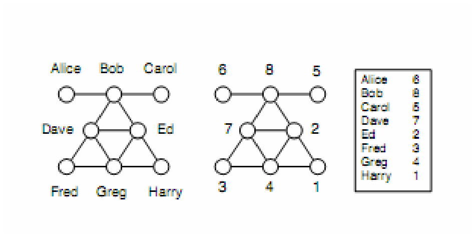
- k-candidate
- k-degree
- k-neighborhood
- k-automorphism

2

k-candidate Anonymity

M Hay et al, *Resisting Structural Re-identification in Anonymized Social Networks* VLDB 2008

G_a the naive anonymization of G through an anonymization mapping f



Alice	6
Bob	8
Carol	5
Dave	7
Ed	2
Fred	3
Greg	4
Harry	1

An individual x ∈ V called the target has a candidate set, denoted cand(x) which consists of the nodes of G_a that could possibly correspond to x

- (background knowledge) structural re-identification where the information of the adversary is about graph structure
- (utility) analysis about structural properties: finding communities, fitting power-law graph models, enumerating motifs, measuring diffusion, accessing resiliency

Closed-World vs Open-World Adversary

Assumption: External information sources are **accurate**, but **not necessarily complete**

- Closed-world: absent facts are false
- Open-world: absent facts are simply unknown

- Adversary may be part of the network
- (locality) Adversary knowledge about a targeted individual tends to be local to the targeted nodes

5

Anonymity through Structural Similarity

[automorphic equivalence]. Two nodes x, y ∈ V are automorphically equivalent (denoted x ≡ y) if there exists an isomorphism from the graph onto itself that maps x to y.

Automorphic equivalence induces a partitioning on V into sets whose members have identical structural properties.

An adversary —even with exhaustive knowledge of the structural position of a target node — cannot identify an individual beyond the set of entities to which it is automorphically equivalent.

6

Adversary Knowledge (model)

An adversary access a source that provides answers to a restricted knowledge query Q evaluated for a single target node of the original graph G.

knowledge gathered by the adversary is accurate.

For target x, use Q(x) to refine the candidate set.

[CANDIDATE SET UNDER Q]. For a query Q over a graph, the candidate set of x w.r.t Q is candQ(x) = {y $\in V_a$ | Q(x) = Q(y)}.

1. Vertex Refinement Queries
2. Subgraph Queries
3. Hub Fingerprint Queries

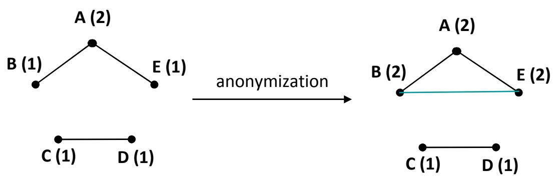
k-degree Anonymity

K. Liu and E. Terzi, *Towards Identity Anonymization on Graphs,* SIGMOD 2008

Privacy model

k-degree anonymity A graph ***G(V, E)*** is ***k****-degree anonymous* if every node in ***V*** has the same degree as ***k-1*** other nodes in ***V***.

Given a graph *G(V, E)* and an integer *k*, modify G via a set of edge addition or deletion operations to construct a new graph *k-degree anonymous* graph G’ in which every node u has the same degree with at least *k*-1 other nodes



[Properties] It prevents the re-identification of individuals by adversaries with *a priori* knowledge of the degree of certain nodes.

9

Degree-sequence anonymization

[k-anonymous sequence] A sequence of integers *d* is *k-anonymous* if every distinct element value in *d* appears at least *k* times.

[100,100, 100, 98, 98,15,15,15]

10

Graph Anonymization algorithm

Two steps

Input: Graph ***G*** with degree sequence ***d***, integer ***k***
Output: ***k***-degree anonymous graph ***G'***

[STEP 1: **Degree Sequence Anonymization**]:
Construct an (optimal) k-anonymous degree sequence ***d'*** from the original degree sequence ***d***

[STEP 2: **Graph Construction**]:
[**Construct**]: Given degree sequence ***d'***, construct a new graph ***G^0(V, E^0)*** such that the degree sequence of ***G^0*** is ***d'***
*[**Transform**]: Transform **G^0(V, E^0)** to **G'(V, E')** so that **SymDiff**(G',G) is minimized.*

Experiments

- **Datasets:**
  - Co-authors (7995 authors of papers in db and theory conference),
  - Enron emails (151 users, edge if at least 5 times),
  - powergrid (generators, transformers and substations in a powergrid network, edges represent high-voltage transmission lines between them),
  - Erdos-Renyi (random graphs with nodes randomly connected to each other with probability p),
  - small-world large clustering coefficient (average fraction of pair of neighbors of a node that are also neighbors) and small average path length (average length of the shortest path between all pairs of reachable nodes),
  - power-law or scale graphs (the probability that a node has degree d is proportional to $d^{-\gamma}$, $\gamma = 2, 3$)

- **Goal (Utility):** degree-anonymization does not destroy the structure of the graph
  - Average path length
  - Clustering coefficient
  - Exponent of power-law distribution

k-neighborhood Anonymity

B. Zhou and J. Pei, *Preserving Privacy in Social Networks Against Neighborhood Attacks,* ICDE 2008

13

1-neighborhood attacks

The neighborhood of $u \in V(G)$ is the induced subgraph of the neighbors of u, denoted by $Neighbor_G(U) = G(N_u)$ where $N_u = \{v \mid (u,v) \in E(G)\}$.

14

Graph Model

Graph G= (V, E, L, F),

V is a set of vertices,

E $\subseteq$ Vx V is a set of edges,

L is a set of labels, and

F a labeling function F: V $\rightarrow$L assigns each vertex a label.

edges do not carry labels

Items in L form a hierarchy.

* $\in$ L -> most general category generalizing all labels.

Given a graph = (V_H, E_H, L, F) and a social network G = (V, E, L, L), an instance of H in G is a tuple (H', f) where H' = ($V_{H'}$,$E_{H'}$,L, F) is a subgraph in G and f: $V_H \rightarrow V_{H'}$, is a bijection function such that

(1) for any u $\in$ V_H, F(f(u)) $\leq$ F(u), /* the corresponding labels in H' are more general */ and

(2) (u, v) $\in$ E_H if and only if (f (u), f(v)) $\in$ $E_{H'}$.

15

G -> G' through a bijection (isomorphism) A

[*k*-neighborhood anonymity] A vertex u $\in$ V (G), u is k anonymous in G' if there are at least (k − 1) other vertices u_1, . . . , u_{k-1} $\in$ V (G) such that $\text{Neighbor}_{G'}(A(u))$, $\text{Neighbor}_{G'}(A(u_1))$, . . ., $\text{Neighbor}_{G'}(A(u_{k-1}))$ are isomorphic.

G′ is k-anonymous if every vertex in G′ is k-anonymous.

Property 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 .

16

Given a social network G, the k-anonymity problem is to compute an anonymization G′ such that

(1) G′ is *k*-anonymous;
(2) each vertex in G is anonymized to a vertex in G′ and G′ does not contain any fake vertex; (no node addition)
(3) every edge in G is retained in G′; and (no node deletion)
(4) the number of edges to be added is minimized.

Utility

Aggregate queries:
compute the aggregate on some paths or subsgraphs satisfying some given conditions

Heuristically, when the number of edges added is as small as possible, G′ can be used to answer aggregate network queries accurately

17

Anonymization Method

Two steps:

STEP 1
Extract the neighborhoods of all vertices in the network
Encode the neighborhood of each node (to facilitate the comparison between neigborhoods)

STEP 2
Greedily, organize vertices into groups and anonymize the neighborhoods of vertices in the same group

18

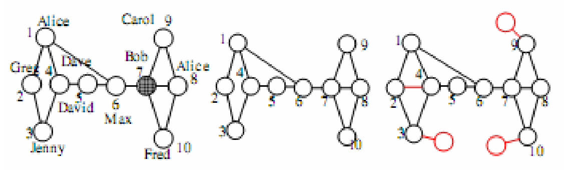
k-Automorphism

L. Zhu, L. Chen and M. Tamer Ozsu, *k-automorphism: a general framework for privacy preserving network publication,* PVLDB 2009

K-Automorphism

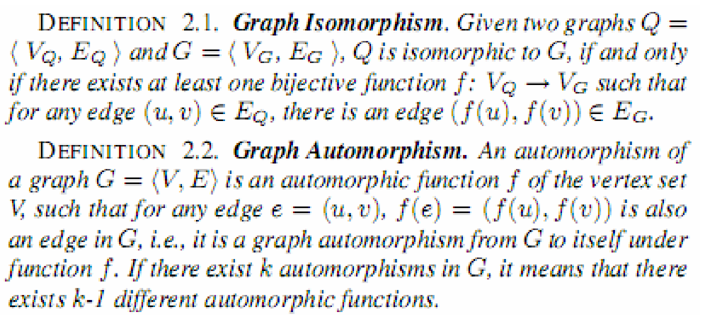
Considers any subgraph query - any structural attack

At least k symmetric vertices no structural differences



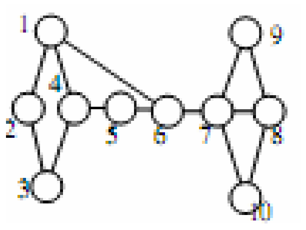
(a) Original Network G (b) Naïve Anonymized Network G' (c) Another Anonymized Network $\hat{G}$

K-Automorphism



DEFINITION 2.1. ***Graph Isomorphism***. *Given two graphs $Q = \langle V_Q, E_Q \rangle$ and $G = \langle V_G, E_G \rangle$, Q is isomorphic to G, if and only if there exists at least one bijective function $f: V_Q \rightarrow V_G$ such that for any edge $(u, v) \in E_Q$, there is an edge $(f(u), f(v)) \in E_G$.*

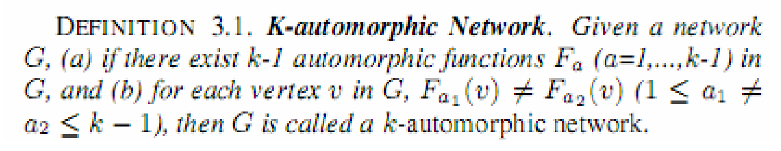
DEFINITION 2.2. ***Graph Automorphism.*** *An automorphism of a graph $G = \langle V, E \rangle$ is an automorphic function f of the vertex set V, such that for any edge $e = (u, v)$, $f(e) = (f(u), f(v))$ is also an edge in G, i.e., it is a graph automorphism from G to itself under function f. If there exist k automorphisms in G, it means that there exists k-1 different automorphic functions.*



map each node of graph G to (another) node of graph G

21

K-Automorphism

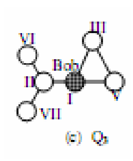


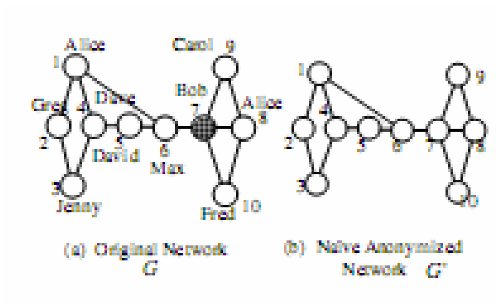
DEFINITION 3.1. ***K-automorphic Network***. *Given a network G, (a) if there exist k-1 automorphic functions F_a (a=1,...,k-1) in G, and (b) for each vertex v in G, $F_{a_1}(v) \neq F_{a_2}(v)$ $(1 \leq a_1 \neq a_2 \leq k-1)$, then G is called a* k-automorphic network.

any k-1 automorphic functions?

22

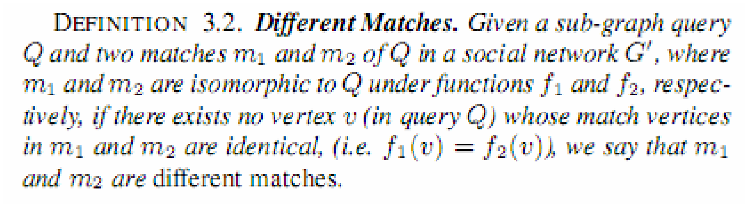
K-Automorphism



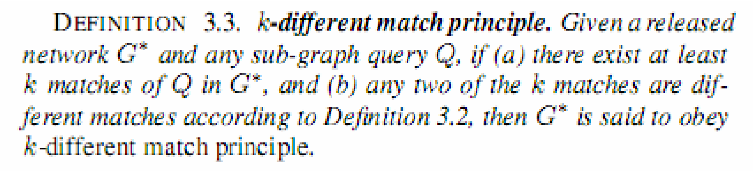


There are eight matches of Q3 in the social network G'
(7; 6; 9; 8; 1; 5), (7; 6; 9; 8; 5; 1), (7; 6; 8; 9; 1; 5), (7; 6; 8; 9; 5; 1),
(7; 6; 10; 8; 1; 5), (7; 6; 10 ; 8; 5; 1), (7; 6; 8; 10; 1; 5) and (7; 6; 8;
10; 5; 1).

K-Automorphism

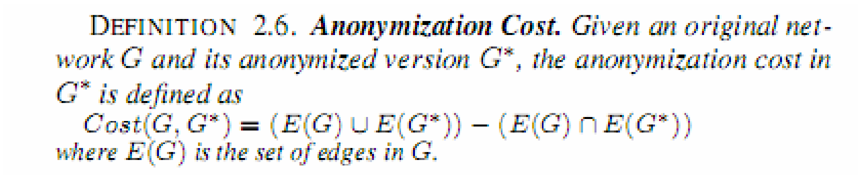


DEFINITION 3.2. ***Different Matches.*** *Given a sub-graph query Q and two matches m_1 and m_2 of Q in a social network G', where m_1 and m_2 are isomorphic to Q under functions f_1 and f_2, respectively, if there exists no vertex v (in query Q) whose match vertices in m_1 and m_2 are identical, (i.e. $f_1(v) = f_2(v)$), we say that m_1 and m_2 are* different matches.



DEFINITION 3.3. ***k-different match principle.*** *Given a released network G^* and any sub-graph query Q, if (a) there exist at least k matches of Q in G^*, and (b) any two of the k matches are different matches according to Definition 3.2, then G^* is said to obey* k-different match principle.

K-Automorphism: Cost



DEFINITION 2.6. ***Anonymization Cost.*** *Given an original network G and its anonymized version G^*, the anonymization cost in G^* is defined as*

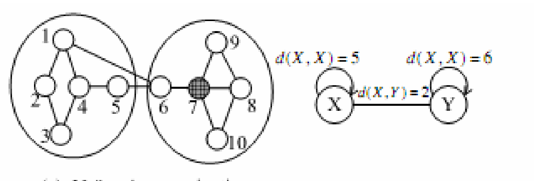
$$Cost(G, G^*) = (E(G) \cup E(G^*)) - (E(G) \cap E(G^*))$$

where $E(G)$ is the set of edges in G.

25

K-Automorphism: Algorithm

compare with Hay et al

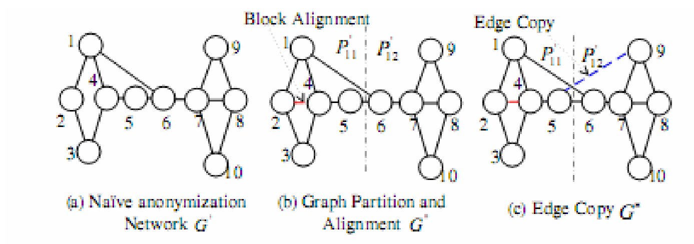


(a) Naïve Anonymization Network G'

(b) Generalized Network

26

K-Match (KM) Algorithm


Block Alignment
Edge Copy
P'_{11}
P'_{12}
1
2
3
4
5
6
7
8
9
10
(a) Naïve anonymization Network G'
(b) Graph Partition and Alignment G''
(c) Edge Copy G^*

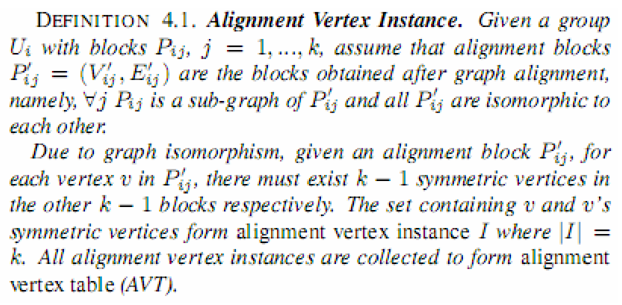
Step 1: Partition the original network into n blocks and cluster these blocks into m roups of at least k blocks

Step 2: Align the blocks into each group to attain isomorphic blocks (add edge (2,4))

Step 3: Apply the "edge-copy" technique to handle matches that cross the two blocks

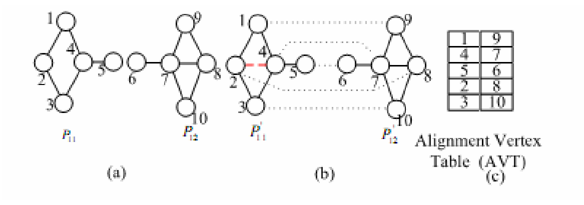
27

K-Match (KM) Algorithm Step 2: Alignment


DEFINITION 4.1. ***Alignment Vertex Instance.*** *Given a group U_i with blocks P_{ij}, $j = 1, \ldots, k$, assume that alignment blocks $P'_{ij} = (V'_{ij}, E'_{ij})$ are the blocks obtained after graph alignment, namely, $\forall j$ P_{ij} is a sub-graph of P'_{ij} and all P'_{ij} are isomorphic to each other.*

Due to graph isomorphism, given an alignment block P'_{ij}, for each vertex v in P'_{ij}, there must exist $k-1$ symmetric vertices in the other $k-1$ blocks respectively. The set containing v and v's symmetric vertices form alignment vertex instance *I where $|I| = k$. All alignment vertex instances are collected to form* alignment vertex table *(AVT).*

Allignment Table

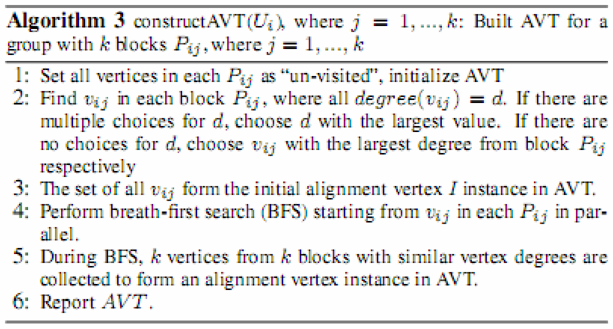


1	9
4	7
5	6
2	8
3	10

28

K-Match (KM) Algorithm: Alignment

Heuristic for finding a good alignment



Algorithm 3 constructAVT(U_i), where $j = 1, \dots, k$: Built AVT for a group with k blocks P_{ij}, where $j = 1, \dots, k$

```
1: Set all vertices in each $P_{ij}$ as "un-visited", initialize AVT
2: Find $v_{ij}$ in each block $P_{ij}$, where all $degree(v_{ij}) = d$. If there are multiple choices for $d$, choose $d$ with the largest value. If there are no choices for $d$, choose $v_{ij}$ with the largest degree from block $P_{ij}$ respectively
3: The set of all $v_{ij}$ form the initial alignment vertex $I$ instance in AVT.
4: Perform breath-first search (BFS) starting from $v_{ij}$ in each $P_{ij}$ in parallel.
5: During BFS, $k$ vertices from $k$ blocks with similar vertex degrees are collected to form an alignment vertex instance in AVT.
6: Report $AVT$.
```

Find k vertices with the same vertex degree

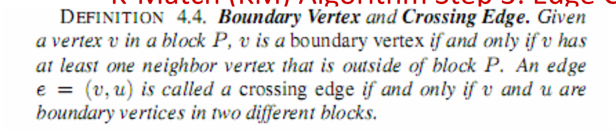
- If many, start with those with high degree
- If none, choose the one with the largest degree

This set -> initial alignment

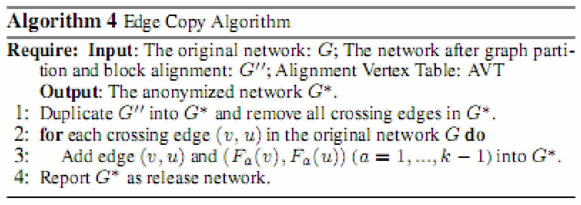
BFS in each block in parallel,

- pairing nodes with similar degree (if there is no corresponding vertex, introduce dummy with the same label as the corresponding)

K-Match (KM) Algorithm Step 3: Edge Copy

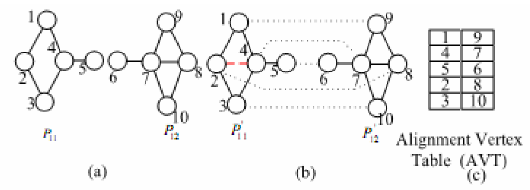


DEFINITION 4.4. ***Boundary Vertex** and **Crossing Edge**. Given a vertex v in a block P, v is a* boundary vertex *if and only if v has at least one neighbor vertex that is outside of block P. An edge $e = (v, u)$ is called a* crossing edge *if and only if v and u are boundary vertices in two different blocks.*



Algorithm 4 Edge Copy Algorithm

```
Require: Input: The original network: $G$; The network after graph partition and block alignment: $G''$; Alignment Vertex Table: AVT
    Output: The anonymized network $G^*$.
1: Duplicate $G''$ into $G^*$ and remove all crossing edges in $G^*$.
2: for each crossing edge $(v, u)$ in the original network $G$ do
3:     Add edge $(v, u)$ and $(F_a(v), F_a(u))$ $(a = 1, \dots, k-1)$ into $G^*$.
4: Report $G^*$ as release network.
```



1	9
4	7
5	6
2	8
3	10

Duplicate crossing edges using the AVT if needed

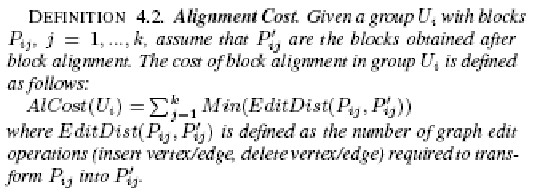
For (1, 6), add (F(1), F(6))

K-Match (KM) Algorithm Step 1: Graph Partitioning

How many blocks so that a small number of edges is added?

Few -> fewer crossing edges, but larger groups (more edges for aligning)

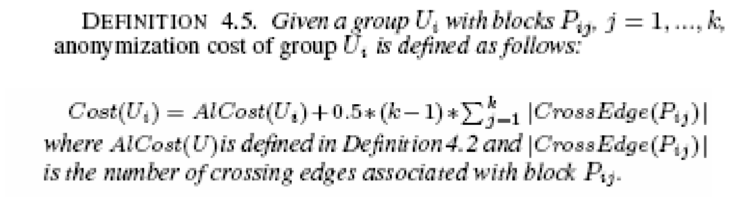
NP complete -> heuristics



DEFINITION 4.2. ***Alignment Cost.*** *Given a group U_i with blocks P_{ij}, $j = 1, ..., k$, assume that P'_{ij} are the blocks obtained after block alignment. The cost of block alignment in group U_i is defined as follows:*

$$AlCost(U_i) = \sum_{j=1}^{k} Min(EditDist(P_{ij}, P'_{ij}))$$

where $EditDist(P_{ij}, P'_{ij})$ is defined as the number of graph edit operations (insert vertex/edge, delete vertex/edge) required to transform P_{ij} into P'_{ij}.



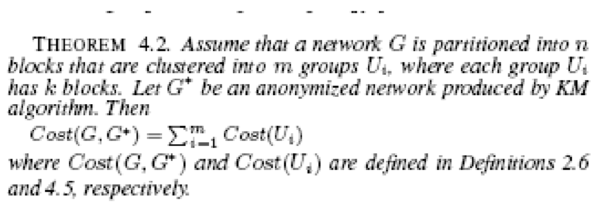
DEFINITION 4.5. *Given a group U_i with blocks P_{ij}, $j = 1, ..., k$,* anonymization cost of group U_i *is defined as follows:*

$$Cost(U_i) = AlCost(U_i) + 0.5 * (k-1) * \sum_{j=1}^{k} |CrossEdge(P_{ij})|$$

where $AlCost(U)$ is defined in Definition 4.2 and $|CrossEdge(P_{ij})|$ is the number of crossing edges associated with block P_{ij}.

31

K-Match (KM) Algorithm Step 1: Graph Partitioning



THEOREM 4.2. *Assume that a network G is partitioned into n blocks that are clustered into m groups U_i, where each group U_i has k blocks. Let G^* be an anonymized network produced by KM algorithm. Then*

$$Cost(G, G^*) = \sum_{i=1}^{m} Cost(U_i)$$

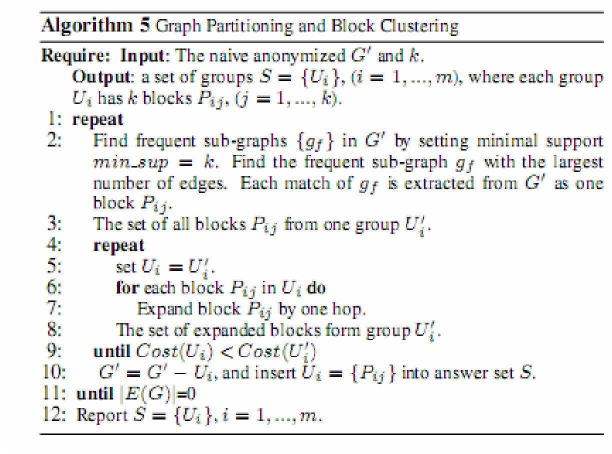
where $Cost(G, G^*)$ and $Cost(U_i)$ are defined in Definitions 2.6 and 4.5, respectively.

32

K-Match (KM) Algorithm Step 3: Graph Partitioning

Find all frequent subgraphs (first group!)

Try to expand them until the cost becomes worst, in which case start a new group

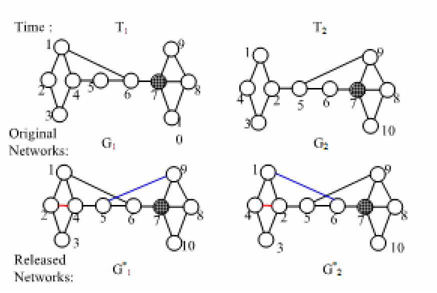


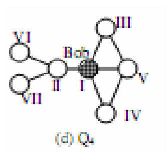
```
Algorithm 5 Graph Partitioning and Block Clustering
Require: Input: The naive anonymized $G'$ and $k$.
    Output: a set of groups $S = \{U_i\}, (i = 1, ..., m)$, where each group
    $U_i$ has $k$ blocks $P_{ij}, (j = 1, ..., k)$.
1:  repeat
2:     Find frequent sub-graphs $\{g_f\}$ in $G'$ by setting minimal support
       $min\_sup = k$. Find the frequent sub-graph $g_f$ with the largest
       number of edges. Each match of $g_f$ is extracted from $G'$ as one
       block $P_{ij}$.
3:     The set of all blocks $P_{ij}$ from one group $U'_i$.
4:     repeat
5:        set $U_i = U'_i$.
6:        for each block $P_{ij}$ in $U_i$ do
7:           Expand block $P_{ij}$ by one hop.
8:        The set of expanded blocks form group $U'_i$.
9:     until $Cost(U_i) < Cost(U'_i)$
10:    $G' = G' - U_i$, and insert $U_i = \{P_{ij}\}$ into answer set $S$.
11: until $|E(G)|$=0
12: Report $S = \{U_i\}, i = 1, ..., m$.
```

33

Dynamic Releases

Example:
Individually satisfy 2-automorphism
Assume that an adversary knows that sub-graph Q4 exists around target Bob at both time T1 and T2.
At time T1, an adversary knows that there are two candidates vertices (2, 7)
Similarly, at time T2, there are still two candidates (4, 7)
Since Bob exists at both T1 and T2, vertex 7 corresponds to Bob





34

Dynamic Releases

Remove all vertex IDs, or permute vertex IDs randomly (so that, a given vertexID does not correspond to the same entity in different publications). Impossible to conduct proper data analysis.

Instead, **vertex ID generalization**

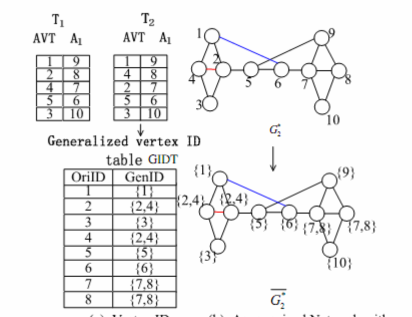
For simplicity, no vertex insertions or deletions in different releases (set of all vertex IDs remains unchanged)

35

Vertex ID Generalization

Given a series of s publications, vertex *v* cannot be identified with a probability higher than 1/k if:

$$Res(v, G_1^*) \cap Res(v, G_2^*) \cap \ldots \cap Res(v, G_s^*) = Res(v, G_1^*)$$
$$where\ |Res(v, G_1^*)| = k.$$



T_1

AVT	A_1
1	9
2	8
4	7
5	6
3	10

T_2

AVT	A_1
1	9
4	8
2	7
5	6
3	10

Generalized vertex ID table GIDT

OriID	GenID
1	{1}
2	{2,4}
3	{3}
4	{2,4}
5	{5}
6	{6}
7	{7,8}
8	{7,8}

(a) Vertex ID Generalization (b) Anonymized Network with Generalized Vertex IDs

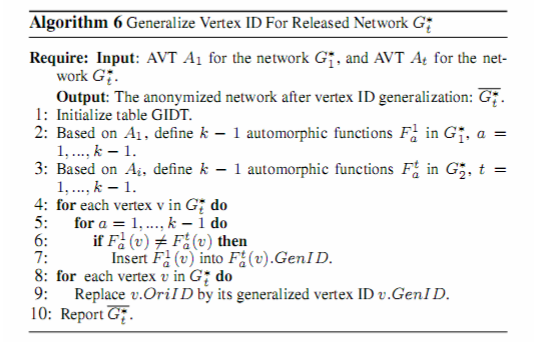
$Res(7, G_1^*) = \{4, 7\}$ and $Res(7, G_2^*) = \{2, 7\}$

$Res(7, G_1^*) \cap Res(7, G_2^*) = Res(7, G_1^*) = \{4, 7\}$

$2.GenID = \{2, 4\}$

36

Vertex ID Generalization: Algorithm

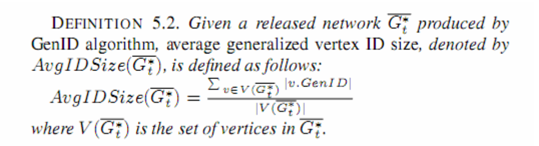


Algorithm 6 Generalize Vertex ID For Released Network G_t^*

```
Require: Input: AVT $A_1$ for the network $G_1^*$, and AVT $A_t$ for the network $G_t^*$.
    Output: The anonymized network after vertex ID generalization: $\overline{G_t^*}$.
 1: Initialize table GIDT.
 2: Based on $A_1$, define $k-1$ automorphic functions $F_a^1$ in $G_1^*$, $a = 1, ..., k-1$.
 3: Based on $A_i$, define $k-1$ automorphic functions $F_a^t$ in $G_2^*$, $t = 1, ..., k-1$.
 4: for each vertex v in $G_t^*$ do
 5:     for $a = 1, ..., k-1$ do
 6:         if $F_a^1(v) \neq F_a^t(v)$ then
 7:             Insert $F_a^1(v)$ into $F_a^t(v).GenID$.
 8: for each vertex $v$ in $G_t^*$ do
 9:     Replace $v.OriID$ by its generalized vertex ID $v.GenID$.
10: Report $\overline{G_t^*}$.
```

37

Vertex ID Generalization: Cost



DEFINITION 5.2. *Given a released network* $\overline{G_t^*}$ *produced by* GenID algorithm, average generalized vertex ID size, *denoted by* $AvgIDSize(\overline{G_t^*})$, *is defined as follows:*

$$AvgIDSize(\overline{G_t^*}) = \frac{\sum_{v \in V(\overline{G_t^*})} |v.GenID|}{|V(\overline{G_t^*})|}$$

where $V(\overline{G_t^*})$ *is the set of vertices in* $\overline{G_t^*}$.

38

Vertex Insertion and Deletion

(Deletion) There is a vertex ID v that exists in G'_1 but not in G'_t
Find an arbitrary vertex ID u that exists in both
Insert v in the generalized vertex ID of u

(Insertion) There is a vertex ID v that exists in G'_t but not in G'_1
Assume that instance I contains v in AVT A_t
For each vertex u in I, insert v in the generalized vertex ID of u --> the new verex exists in at least k vertices in G'_t

Evaluation

Prefuse (129 nodes, 161 edges)
Co-author graph (7995 authors in database and theory, 10055 edges)

Synthetic
Erdos Renyi 1000 nodes
Scale free, $2 < \gamma < 3$

All k = 10 degree anonymous, but no sub-graph anonymous

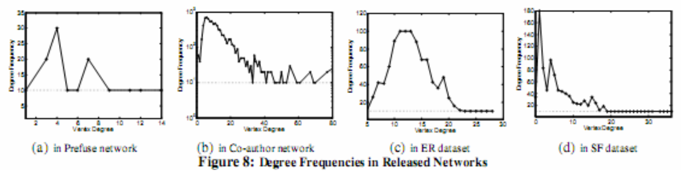


Figure 8: Degree Frequencies in Released Networks

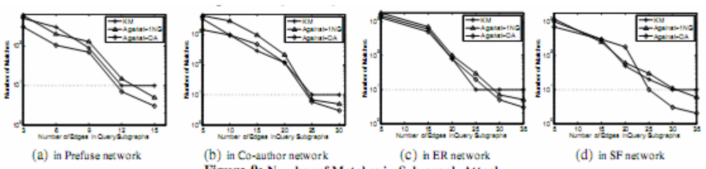


Figure 9: Number of Matches in Sub-graph Attack

Against-Degree

Against-1Neigbhorhood

Generalization

Utility:

Total degree differences

Average shortest path length

Aveage cluster co-efficient

Dynamic Releases:

10% edges -> 1.93 (prefuse) 8.03 (co-author) average generalized vertex ID size

Next: Privacy Attributes

43