k-Symmetry Model: A General Framework To Achieve Identity Anonymization In Social Networks

Wentao Wu

School of Computer Science and Technology,
Fudan University, Shanghai, China

1 Introduction

Social networks represent relationships between individuals when they take part in some social activities. For example, a telephone communication network could be built according to the phoning record from residents in a city. The WWW and MSN communication network are other examples of social networks. Recently, due to the availability of more and more social network datasets, inspecting the properties of these networks has been attracting more and more researchers in different areas such as sociology, physics, and computer science. Previous work has shown that although different social networks are built according to different rules, they share some ubiquitous properties such as *small world* phenomenon [1], the *scale-free* degree distribution [2], *assortative mixing* [3] and *self-similarity* [4]. These properties are quite different compared with the random networks of the same scale, which implies that the real networks are evolving under certain self-organization rules. Deeper studies then demand the publication of more social networks.

However, many social networks contain sensitive information which is relevant to the individuals' personal privacy. If they are published without any preprocessing, or just preprocessed by removing the identifier (i.e., replacing the identifier by a randomly chosen integer, and this strategy is called *naive anonymization*) from each node, there will be privacy leakage problems for the individuals recorded by the network. For instance, Potterat et al. [5] published a social network which shows a set of individuals related by sexual contacts and shared drug injections. While scientists could know more on how HIV spreads by using the information provided in this network, the privacy of the individuals in this network may be possibly comprised due to unknown attacks. In fact, as shown in [6], a simple model named *active attack* could be used to expose considerable amount of relationship information between different individuals with high probability of success and not much overhead, even in very large social networks. As a result, the sensitivity of the data often prevents the data owner from publishing it.

Fortunately, although the active attack is very powerful, the case where an intended adversary would use it is limited since it requires the adversary to do the attack before the network is published. A more natural model named *passive attack* is also mentioned in [6], which simply probes relationship information by re-identifying individuals in the network according to some background *structural* knowledge which is unique in the published network with respect to the targeted nodes. As pointed out in [6], this method may even be used by an ordinary user of the network who is just a bit more curious about his neighbors and thus could hardly be condemned morally.

In this paper, we focus on the problem of resisting re-identification of individuals from naively anonymized social networks through structural background knowledge. We generalize the attack model described in [7] and propose a new anonymization strategy.

We also analyze the utility problem of the network after anonymization in detail and propose a sampling method to achieve good approximation of the statistical properties of the original network. Extensive study through a series of experimental evaluations demonstrates the effectiveness of the sampling method.

The rest of the paper is organized as follows: Section 2 gives our attack model which is a generalization of the model proposed in [7] and also introduces basic concepts and notations that will be used throughout the paper; Section 3 describes the anonymization strategy, with detailed analysis. The utility problem is also studied here, with both theoretical and experimental examinations; Section 4 further describes two variants of the basic model proposed in Section 3, by introducing more flexibilities; Section 5 discusses two related problems and gives some remarks; Section 6 summarizes related work; Section 7 concludes the paper and points out future work.

2 Preliminaries

In this section, we first introduce basic concepts and notations that will be used throughout this paper and then formalize the attack model.

2.1 Basic Concepts And Notations

An unlabeled, unweighed simple graph G can be represented as $G = (V(G), E(G))$, where $V(G)$ is the set of vertices and $E(G) \subseteq V(G) \times V(G)$ is the set of edges. G is said to be *undirected* if we don't distinguish the edge (u, v) from (v, u), where $u, v \in V(G)$. If u is connected to v through an edge (u, v), then u is called a *neighbor* of v. We use $N(v)$ to denote the set of all neighbors of v. The *degree* of a vertex v is then defined to be $|N(v)|$. A vertex sequence $v_1, v_2, ..., v_n$ of $V(G)$ is a *path* in G if $v_i \neq v_j (i \neq j)$ and $(v_i, v_{i+1}) \in E(G)$. G is *connected* if for any $u, v \in V(G)$, there is a path between u and v. Formally and without loss of generality, we now model a social network as an unlabeled, unweighed, undirected and connected simple graph.

A *subgraph* S of a graph G is a graph whose set of vertices and set of edges are all subsets of G. A (vertex) *induced subgraph* is one that consists of some of the vertices of the original graph and all of the edges that connect them in the original.

A set of subsets of $V(G)$, $\mathcal{V} = \{V_1, ..., V_m\}$, is said to be a *partition* of $V(G)$, if $\cup_{i=1}^{m} V_i = V(G)$ and $V_i \cap V_j = \emptyset, \forall i \neq j$. The elements of $\mathcal{V}$ are also called *cell*s. A *trivial cell* is the cell with cardinality one. If every cell of a partition is trivial, then the partition is a *discrete partition.* $\mathcal{V}$ is said to be *equitable* if $\forall i, j \in \{1, 2, ..., m\}$, the numbers $|N(v) \cap V_j|$ are constant on each cell, i.e. depend on the cell index i only and not on the vertex $v \in V_i$. So for an equitable partition $\mathcal{V}$, $|\mathcal{V}| = m$, we can define a $m \times m$ cell-level adjacency matrix $CAdjM(\mathcal{V})$ such that $CAdjM(\mathcal{V})[i, j] = |N(v) \cap V_j|, \forall v \in V_i$. Given two partitions $\mathcal{V}_1$, $\mathcal{V}_2$ of $V(G)$, $\mathcal{V}_1$ is *finer* than $\mathcal{V}_2$ (denoted as $\mathcal{V}_1 \preccurlyeq \mathcal{V}_2$), if for any cell $V \in \mathcal{V}_1$, there exists some cell $U \in \mathcal{V}_2$ such that $V \subseteq U$. $\mathcal{V}_2$ is then called *coarser* than $\mathcal{V}_1$.

Let S be any finite set. A *permutation* on S is defined as a bijection $\pi : S \to S$. We use $\pi(s)$ or s^π to denote the image of s under π, for any $s \in S$. Now let π be a permutation on $V(G)$, and $V = \{v_1, v_2, ..., v_n\} \subseteq V(G)$ be any subset of $V(G)$. We then define the image of V under π to be $V^\pi = \{v_1^\pi, v_2^\pi, ..., v_n^\pi\}$. Next let $\mathcal{V} = \{V_1, V_2, ..., V_m\}$ be any partition of $V(G)$, then similarly, we define the image of $\mathcal{V}$ under π to be $\mathcal{V}^\pi = \{V_1^\pi, V_2^\pi, ..., V_m^\pi\}$. If $\mathcal{V}^\pi = \mathcal{V}$, then $\mathcal{V}$ is said to be *invariant* under the action of π.

Now suppose π is a permutation on $V(G)$, and define graph $G^\pi = (V(G)^\pi, E(G)^\pi)$, where $E(G)^\pi = \{(u^\pi, v^\pi)|(u,v) \in E(G)\}$. An *automorphism* of G is a permutation π on $V(G)$ such that $G^\pi = G$. It is easy to see that all the automorphisms of G forms a *group*, namely, the *automorphism group* of G. We denote it as $Aut(G)$.

Based on the definition of $Aut(G)$, we now introduce a relation R on $V(G)$ such that $\forall u, v \in V(G)$, uRv if and only if there exists some automorphism $g \in Aut(G)$ such that $u^g = v$. It is easy to check that R is an equivalence relation by noticing the fact that $Aut(G)$ is a group. R then induces a partition on $V(G)$, namely, the *automorphism partition.* Each cell of the automorphism partition is also called as an *orbit* of the automorphism group $Aut(G)$. We hence denote the automorphism partition of G as $Orb(G)$. It's clear that $Orb(G)$ is equitable. Furthermore, $Orb(G)$ is invariant under the action of any $g \in Aut(G)$.

Two graphs G and G' are said to be *isomorphic* if there is a bijection f between $V(G)$ and $V(G')$ such that $(u,v) \in E(G)$ if and only if $(f(u), f(v)) \in E(G')$. f is then called an *isomorphism* mapping between G and G'. We denote $G \cong G'$ if G and G' are isomorphic.

2.2 The Attack Model

Hay et al. [7] proposed the concept of *candidate set* as follows. Let G be a social network, G_a be the corresponding graph after naive anonymization, and $v \in V(G_a)$ be an individual in the network, called the *target.* Suppose P is some background structural knowledge about v. Then the candidate set of v is defined as $C(v) = \{u|u \in V(G_a) \wedge P(u) = P(v)\}$. Obviously, the larger $C(v)$ is , the harder that v could be re-identified from G_a. In particular, the target v could be definitely re-identified from G_a by using the background structural knowledge P if and only if $|C(v)| = 1$.

Example 1 (Candidate Set). Figure 1 gives a network G and its naively anonymized version G_a. Suppose the background structural knowledge of Bob is that *Bob has 4 neighbors*, then $C(Bob) = \{2, 4, 7, 8\}$. Now suppose the background structural knowledge of Bob becomes that *Bob has 2 neighbors with degree 1*, then $C(Bob) = \{8\}$ and thus Bob is re-identified from G_a in this case.

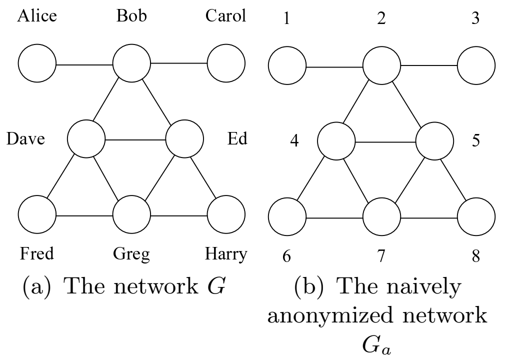


Fig. 1. Illustration of the candidate set.

The attack model proposed in [7] is then by restricting the background structural knowledge P to different levels of neighborhood information of the target, where the radius of the neighborhood is measured by the distance (i.e. shortest path length) of the farthest vertices in the neighborhood from the target. Background structural knowledge involving larger neighborhood then will lead to a possibly smaller candidate set. Our attack model generalizes this by dropping the above restriction to allow P to be any background structural knowledge of a target individual. Based on the definition of automorphism partition, we see that two vertices in the network G_a are *structurally equivalent* if and only if they belong to the same orbit of $Orb(G_a)$. In other words, suppose $Orb(G_a) = \{V_1, V_2, ..., V_m\}$, and the target $v \in V_i$, then $|C_v| \geq |V_i|$, no matter which background structural knowledge P is used. So if we could modify G_a so that each $V_i \in Orb(G_a)$ contain at least k vertices, where k is a specified threshold to control the anonymization granularity, then we have $|C_v| \geq k$ and thus the probability of any target individual being re-identified from the network would be at most $\frac{1}{k}$. Based on this intuition, we propose the k-symmetry model in the next section to achieve the anonymity of the network. Since $G \cong G_a$, for notational convenience, we use G instead of G_a in the rest of this paper.

3 The *k*-Symmetry Model

This section gives formal descriptions of our k-symmetry model. We formalize the motivation in 3.1 which has been shortly mentioned in the previous part. We then introduce an operation in 3.2 which is important in the description of the model. We also set up several key properties of the introduced operation which seem to be useful when analyzing the anonymization procedure. 3.3 sets up the model formally and 3.4 discusses the utility problem in detail. Finally, 3.5 gives experimental results which demonstrate the effect of the sampling method used to preserve the utility of the original network.

3.1 Motivation

Intuitively, if two vertices u and v belong to the same orbit of $Orb(G)$, then they cannot be distinguished by any structure-based properties. We would then say u and v are *structurally equivalent*. Therefore, if the size of each orbit in the automorphism partition is at least k, then each vertex of G will have at least $k-1$ structurally equivalent counterparts. As a result, to distinguish any individual in the social network, just using any structure-based background knowledge will suffer a cost of $k-1$. Or equivalently, the success probability of such attacks are at most $\frac{1}{k}$.

We now give a formal definition to the above intuition.

Definition 1. *Given a graph G and an integer k, if $\forall V_i \in Orb(G)$, $|V_i| \geq k$, then G is called **k-symmetric**. The requirement that $\forall V_i \in Orb(G)$, $|V_i| \geq k$ is called the **k-symmetry** constraint.*

Then the problem becomes: *Given a graph G and an integer k, how to modify G so that G is k-symmetric?* In this paper, we will only consider vertex/edge addition as the graph modification operations. As a result, the original graph G then must be a subgraph of the anonymized graph G'.

3.2 The Orbit Copying Operation

The basic idea to make the given graph G *k-symmetric* is to enlarge each orbit in $Orb(G)$ until its size is larger than k. The key point here is then how to do the enlargement so that the vertices in the augmented orbit are still structurally equivalent. In this section, we introduce an operation which we named as *orbit copying*, and we will show that the resulted graph by applying this operation will satisfy the above requirement.

Definition 2 (Sub-automorphism partition). *Let G be a graph, $Aut(G)$ be its automorphism group and $Orb(G)$ be its automorphism partition. Suppose a vertex partition $\mathcal{V}$ of $V(G)$ satisfies the condition: $\forall O \in \mathcal{V}$, and $\forall u, v \in O$, $\exists g \in Aut(G)$ such that $u^g = v$ and $\mathcal{V}^g = \mathcal{V}$. Then $\mathcal{V}$ is called a* ***sub-automorphism partition*** *of G.*

Clearly, if $\mathcal{V}$ is a sub-automorphism partition of G, then $\mathcal{V} \preccurlyeq Orb(G)$. Furthermore, $\mathcal{V}$ is also equitable. In particular, $Orb(G)$ is also a sub-automorphism partition of G.

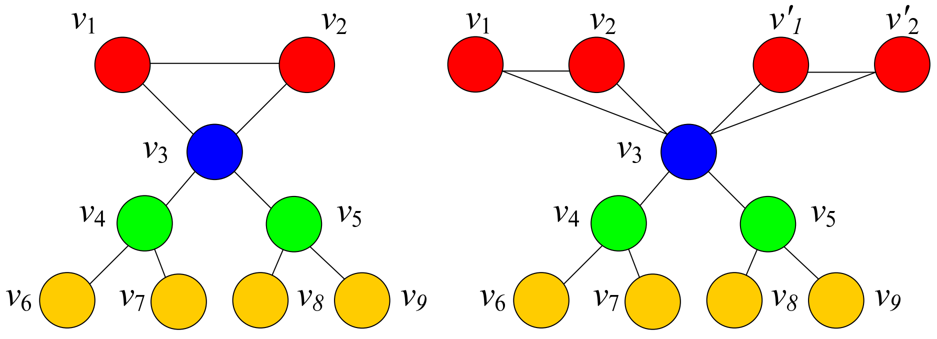
Definition 3 (Orbit Copying). *Given a graph G and a sub-automorphism partition $\mathcal{V}$ of G. Suppose $V \in \mathcal{V}$, an* ***orbit copying*** *operation $Ocp(G, \mathcal{V}, V)$ is defined as follows:*

For each $v \in V$, introduce a new vertex v' into graph G and:

1. *if $(u, v) \in E(G)$, $u \in U$, $U \in \mathcal{V}$ and $U \neq V$, then add an edge (u, v') into G;*
2. *if $(u, v) \in E(G)$, $u \in V$, then add an edge (u', v') into G.*

Note that, in the resulted graph G' after an orbit copying operation $Ocp(G, \mathcal{V}, V)$, the vertex set V and its copied counterpart V' cannot be distinguished from each other. Therefore, we say that the corresponding vertices v and v' could be *copied to* each other and this relation is symmetric.

Example 2. As shown in Fig 2(a), the original graph G has the automorphism partition $Orb(G) = \{V_1, V_2, V_3, V_4\}$, where $V_1 = \{v_1, v_2\}$, $V_2 = \{v_3\}$, $V_3 = \{v_4, v_5\}$, and $V_4 = \{v_6, v_7, v_8, v_9\}$. Fig 2(b) shows the graph after the orbit V_2 is copied.



(a) The original graph G (b) The graph after orbit $\{v_1, v_2\}$ is copied

Fig. 2. Illustration of the orbit copying operation: the vertex partition of G is exactly $Orb(G)$, and vertices in the same orbit are colored by the same color.

We next show an important theorem which will be the foundation of the anonymization procedure introduced in the next section. But before that, we shall first prove several lemmas.

The first lemma (Lemma 1) states that a single orbit copying operation $Ocp(G, \mathcal{V}, V)$ will obtain a sub-automorphism partition of the resulted graph.

Lemma 1. *Let G be any given graph, and $\mathcal{V}$ is any sub-automorphism partition of G. Suppose $\mathcal{V} = \{V_1, V_2, ..., V_m\}$, and the vertex partition after applying an orbit copying* *operation $Ocp(G, \mathcal{V}, V_i)$ is $\mathcal{V}'_i = \{V_1, V_2, ..., V'_i, ..., V_m\}$. Now let G'_i be the corresponding graph after the orbit copying operation on V_i, then $\mathcal{V}'_i$ is a sub-automorphism partition of G'_i, $\forall 1 \leq i \leq m$.*

Proof. We need to show that, for each cell $V \in \mathcal{V}'_i$, and $\forall u, v \in V$, there is an $h \in Aut(G'_i)$ such that $u^h = v$ and $(\mathcal{V}'_i)^h = \mathcal{V}'_i$. We first consider the case that $V = V'_i$.

Suppose $V_i = \{v_1, v_2, ..., v_s\}$, then the vertices in V'_i could be represented as $V'_i = \{v_1, v_2, ..., v_s, v'_1, v'_2, ..., v'_s\}$. For notational convenience, we use W_i to denote the set $\{v_1, v_2, ..., v_s\}$ (actually, $W_i = V_i$), and use W'_i to denote the set $\{v'_1, v'_2, ..., v'_s\}$, which lead to $V'_i = W_i \cup W'_i$ and $W_i \cap W'_i = \emptyset$ under this representation. Namely, W_i and W'_i forms a partition of V'_i, and furthermore, $V(G'_i) = V(G) \cup W'_i$. We then define a mapping f, such that $f : W_i \to W'_i$, where $f(v_j) = v'_j$, for $1 \leq j \leq s$. It's clear that f is a bijection, and we use f^{-1} to denote its inverse mapping.

Now suppose u and v are two arbitrary vertices in V'_i. Without loss of generality, there are two cases we need to follow, namely, whether u and v are both in W_i, or u and v are in W_i and W'_i respectively. We next prove these two cases one by one. The idea is that in each case we shall construct an $h \in Aut(G'_i)$ such that $u^h = v$ and $(\mathcal{V}'_i)^h = \mathcal{V}'_i$.

Case 1 *Both u and v belong to W_i.*

Since $W_i = V_i$, there is a $g \in Aut(G)$ such that $u^g = v$ and $\mathcal{V}^g = \mathcal{V}$, due to the fact $\mathcal{V}$ is a sub-automorphism partition of $V(G)$. Now construct a new permutation h on $V(G')$ such that h acts in the same way as g on $V(G)$ and $(v'_j)^h = f(v_j^g)$ (remember that $v_j^g \in W_i = V_i$), for $1 \leq j \leq s$. Clearly $u^h = u^g = v$. We now show that $h \in Aut(G'_i)$.

Suppose $e = (a, b) \in E(G'_i)$ is an arbitrary edge of G'_i. We need to show that $e^h = (a^h, b^h) \in E(G'_i)$. There are totally three different cases:

(1) $e \in E(G)$. Then $a \in V(G)$ and $b \in V(G)$, and $e^h = (a^h, b^h) = (a^g, b^g) = e^g \in E(G) \subseteq E(G'_i)$.

(2) $e \notin E(G)$, but $a \in V(G)$ and $b \in W'_i$ (the case $b \in V(G)$ and $a \in W'_i$ is the same). Then $e^h = (a^h, b^h) = (a^g, f((f^{-1}(b))^g))$. Since $b \in W'_i$, $f^{-1}(b) \in W_i = V_i$, thus $(a, f^{-1}(b)) \in E(G)$, otherwise $(a, b) \notin E(G'_i)$. Therefore, $(a^g, (f^{-1}(b))^g) \in E(G)$ and what's more, $(f^{-1}(b))^g \in V_i$, since $\mathcal{V}^g = \mathcal{V}$. As a result, $(a^g, f((f^{-1}(b))^g)) \in E(G'_i)$.

(3) $e \notin E(G)$, and both a and b are in W'_i. Then we have $e^h = (a^h, b^h) = (f((f^{-1}(a))^g), f((f^{-1}(b))^g))$. Since $f^{-1}(a) \in V_i$ and $f^{-1}(b) \in V_i$, we must have $(f^{-1}(a), f^{-1}(b)) \in E(G)$, otherwise $(a, b) \notin E(G'_i)$. Thus, we have $((f^{-1}(a))^g, (f^{-1}(b))^g) \in E(G)$, and then $(f((f^{-1}(a))^g), f((f^{-1}(b))^g)) \in E(G'_i)$ since both $(f^{-1}(a))^g$ and $(f^{-1}(b))^g$ are in V_i (again due to $\mathcal{V}^g = \mathcal{V}$).

We now prove that $(\mathcal{V}'_i)^h = \mathcal{V}'_i$. But since h acts in the same way as g on $V(G)$, and since $\mathcal{V}^g = \mathcal{V}$, we need only to prove that $(V'_i)^h = V'_i$. According to the definition of h, $(V'_i)^h = \{v_1^h, ..., v_s^h, (v'_1)^h, ..., (v'_s)^h\} = \{v_1^g, ..., v_s^g, f(v_1^g), ..., f(v_s^g)\}$. Since $W_i = V_i = V_i^g = \{v_1^g, ..., v_s^g\}$, and $W'_i = f(W_i) = f(V_i) = f(V_i^g) = \{f(v_1^g), ..., f(v_s^g)\}$, we thus have $(V'_i)^h = W_i \cup W'_i = V'_i$, which completes the proof of **Case** 1.

(**Note**: The case both u and v belong to W' is proved in the same way.)

Case 2 *$u \in W_i$ and $v \in W'_i$.*

Since $v \in W'_i$, we have $f^{-1}(v) \in W_i$. Then there is an automorphism $h_1 \in Aut(G'_i)$ such that $u^{h_1} = f^{-1}(v)$ and $(\mathcal{V}'_i)^{h_1} = \mathcal{V}'_i$, as described in the proof of Case 1. Now construct a permutation h_2 on $V(G'_i)$ such that $v_j^{h_2} = v'_j$ and

$(v'_j)^{h_2} = v_j$, for $1 \leq j \leq s$ and $v^{h_2} = v$ for any $v \in (V(G'_i) \setminus V'_i)$. Let $h = h_1 \cdot h_2$ and we then have $u^h = u^{h_1 h_2} = (u^{h_1})^{h_2} = (f^{-1}(v))^{h_2} = f(f^{-1}(v)) = v$. We next prove that $h \in Aut(G'_i)$ and $(\mathcal{V}'_i)^h = \mathcal{V}'_i$. But since $h_1 \in Aut(G'_i)$ and $(\mathcal{V}'_i)^{h_2} = \mathcal{V}'_i$, if we can show $h_2 \in Aut(G'_i)$ and $(\mathcal{V}'_i)^{h_2} = \mathcal{V}'_i$, then we are done, due to the fact $Aut(G'_i)$ is a group and $(\mathcal{V}'_i)^h = ((\mathcal{V}'_i)^{h_1})^{h_2} = (\mathcal{V}'_i)^{h_2}$. To prove $h_2 \in Aut(G'_i)$, we only need to prove that, for each edge $e = (a, b) \in E(G'_i)$, we must have $e^{h_2} = (a, b)^{h_2} \in E(G'_i)$. Actually, since h_2 just exchanges the corresponding vertices in W_i and W'_i, we only need to consider the edges having at least one end in W_i or W'_i. There are two different cases:

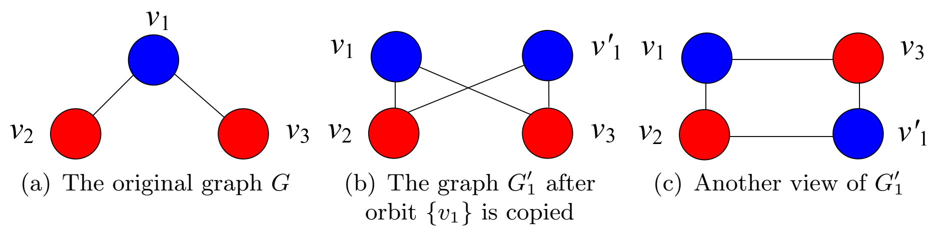
(1) Both a and b are in W_i. Then $e^{h_2} = (a^{h_2}, b^{h_2}) = (f(a), f(b)) \in E(G'_i)$.
(**Note**: The case both a and b are in W'_i is proved in the same way.)

(2) a in W_i and $b \in (V(G'_i) \setminus V'_i)$. Then $e^{h_2} = (a^{h_2}, b^{h_2}) = (f(a), b) \in E(G'_i)$.
(**Note**: The case both a in W'_i and $b \in (V(G'_i) \setminus V'_i)$ is proved in the same way.)

Notice that the case $a \in W_i$ and $b \in W'_i$ (or $a \in W'_i$ and $b \in W_i$) is impossible. Therefore, we now have proved that $h_2 \in Aut(G'_i)$ and hence $h \in Aut(G'_i)$.
Finally, since h_2 fixes vertices in the cells other than V'_i, to prove $(\mathcal{V}'_i)^{h_2} = \mathcal{V}'_i$, we again only need to prove that $(V'_i)^{h_2} = V'_i$. Actually, we have $(V'_i)^{h_2} = \{v_1^{h_2}, ..., v_s^{h_2}, (v'_1)^{h_2}, ..., (v'_s)^{h_2}\} = \{v'_1, ..., v'_s, v_1, ..., v_s\} = V'_i$, which completes the proof of **Case** 2.
(**Note**: The case $u \in W'_i$ and $v \in W_i$ is proved in the same way.)

Suppose now $V \neq V'_i$, then $\forall u, v \in V$, $\exists g \in Aut(G)$ such that $u^g = v$ and $\mathcal{V}^g = \mathcal{V}$. Construct a permutation h on $V(G'_i)$ such that h acts in the same way as g on $V(G)$, and $(v'_j)^h = f(v_j^g)$. As shown in **Case** 1, the h so constructed is an automorphism of G'_i satisfying $(\mathcal{V}'_i)^h = \mathcal{V}'_i$, and clearly $u^h = u^g = v$. This completes the whole proof.

Remark 1. According to Lemma 1, given a graph G and its automorphism partition $Orb(G)$, an orbit copying operation $Ocp(G, Orb(G), V_i)$ on G will result a sub-automorphism partition $\mathcal{V}'_i$ of the resulted graph G'_i. Sometimes $\mathcal{V}'_i = Orb(G'_i)$ (as the case shown in Fig 2) but sometimes it is not the case. Fig 3 shows a trivial counterexample. Here, $Orb(G) = \{\{v_1\}, \{v_2, v_3\}\}$ and $\mathcal{V}'_1 = \{\{v_1, v'_1\}, \{v_2, v_3\}\}$. But if we take another view of G'_1 (as in Fig 3(c)), it's easy to see that all the four vertices of G'_1 must belong to the same orbit of G'_1, namely, $Orb(G'_1) = \{\{v_1, v'_1, v_2, v_3\}\}$ and therefore, $\mathcal{V}'_1 \neq Orb(G'_1)$.



(a) The original graph G (b) The graph G'_1 after orbit $\{v_1\}$ is copied (c) Another view of G'_1

Fig. 3. A trivial counterexample that $\mathcal{V}'_i \neq Orb(G'_i)$

The next lemma (Lemma 2) generalizes the result of Lemma 1, which says that if we successively apply the orbit copying operations on the same cell, the conclusion of Lemma 1 will still hold.

Lemma 2. *Let G be any given graph, and $\mathcal{V}$ be any sub-automorphism partition of G. Suppose $\mathcal{V} = \{V_1, V_2, ..., V_m\}$, and the vertex partition after applying $N \geq 0$ orbit copying operations $Ocp(G, \mathcal{V}, V_i)$ on the same cell V_i is $\mathcal{V}_i^{(N)} = \{V_1, V_2, ..., V_i^{(N)}, ..., V_m\}$. Now let $G_i^{(N)}$ be the corresponding graph, then $\mathcal{V}_i^{(N)}$ is a sub-automorphism partition of $G_i^{(N)}$, $\forall 1 \leq i \leq m$.*

Proof. If $N = 0$, $G_i^{(0)} = G$ and $\mathcal{V}_i^{(0)} = \mathcal{V}$, thus the lemma holds trivially. Now let $N \geq 1$. Suppose $V_i = \{v_1, v_2, ..., v_s\}$ and $V_i^{(N)} = \{v_1, ..., v_s, v_1^{(1)}, ..., v_s^{(1)}, ..., v_1^{(N)}, ..., v_s^{(N)}\}$ where $v_1^{(n)}, ..., v_s^{(n)}$ are vertices copied from $v_1, ..., v_s$ in the n-th operation, $1 \leq n \leq N$. Similarly as in the proof of Lemma 1, we use $W_i^{(0)} = V_i$ to denote the set $\{v_1, ..., v_s\}$, and use $W_i^{(n)}$ to denote the set $\{v_1^{(n)}, ..., v_s^{(n)}\}$. Also, we define a mapping $f_n : W_i^{(0)} \to W_i^{(n)}$ for each $1 \leq n \leq N$, such that $f_n(v_j) = v_j^{(n)}$ where $1 \leq j \leq s$. For convenience, we just define $f_0 : W_i^{(0)} \to W_i^{(0)}$ to be the identical mapping on $W_i^{(0)}$, namely, $f_0(v_j) = v_j$ where $1 \leq j \leq s$. Clearly each f_n is a bijection. To prove the lemma, we again need to show that, for each cell $V \in \mathcal{V}_i^{(N)}$, and $\forall u, v \in V$, there is an $h \in Aut(G_i^{(N)})$ such that $u^h = v$ and $(\mathcal{V}_i^{(N)})^h = \mathcal{V}_i^{(N)}$. The idea is so similar to the proof of Lemma 1 that in the following process we just give some sketch and not repeat the details again.

We first consider the case $V = V_i^{(N)}$, and there are again two types of situation, namely whether u and v are in the same $W_i^{(j)}$ or not, where $0 \leq j \leq N$. In the first situation, without loss of generality, suppose $u, v \in W_i^{(0)}$. Then there is a $g \in Aut(G)$ such that $u^g = v$ and $\mathcal{V}^g = \mathcal{V}$. We next construct a permutation h on $V(G_i^{(N)})$ such that h acts in the same way as g on $V(G)$, and $(v_j^{(n)})^h = f_n(v_j^g)$, for $1 \leq j \leq s$, and $1 \leq n \leq N$. We then show that $h \in Aut(G_i^{(N)})$ and $(\mathcal{V}_i^{(N)})^h = \mathcal{V}_i^{(N)}$. In the second situation, without loss of generality, suppose $u \in W_i^{(p)}$ and $v \in W_i^{(q)}$, $p \neq q$. Let $u_0 = f_p^{-1}(u)$, $v_0 = f_q^{-1}(v)$, then $u_0 \in W_i^{(0)}$, $v_0 \in W_i^{(0)}$, and there is a $g \in Aut(G)$ such that $u_0^g = v_0$ and $\mathcal{V}^g = \mathcal{V}$. Construct an $h_1 \in Aut(G_i^{(N)})$ as in the first situation. Construct a permutation h_2 on $V(G_i^{(N)})$ such that h_2 just exchanges corresponding vertices in $W_i^{(p)}$ and $W_i^{(q)}$, and fixing any other vertex in $V(G_i^{(N)})$. Formally, $(v_j^{(p)})^{h_2} = v_j^{(q)}$, $(v_j^{(q)})^{h_2} = v_j^{(p)}$, and $v^{h_2} = v$, $\forall v \in V(G_i^{(N)}) \setminus V_i^{(N)}$. We then prove that $h_2 \in Aut(G_i^{(N)})$ and $(\mathcal{V}_i^{(N)})^{h_2} = \mathcal{V}_i^{(N)}$. Now let $h = h_1 \cdot h_2$, then $h \in Aut(G_i^{(N)})$, $(\mathcal{V}_i^{(N)})^h = \mathcal{V}_i^{(N)}$ and $u^h = u^{h_1 h_2} = (u^{h_1})^{h_2} = (f_p(v_0))^{h_2} = f_q(v_0) = v$.

Now consider the case that $V \neq V_i^{(N)}$, then $\forall u, v \in V$, there is a $g \in Aut(G)$ such that $u^g = v$ and $\mathcal{V}^g = \mathcal{V}$. We can construct a $h \in Aut(G_i^{(N)})$ in the same way as we do in the proof of the first situation of the case $V = V_i^{(N)}$, which gives $u^h = u^g = v$ and $(\mathcal{V}_i^{(N)})^h = \mathcal{V}_i^{(N)}$. This completes the whole proof.

The next two lemmas (Lemma 3 and 4) claim that if we rearrange the order of operations in a given orbit copying operation sequence, the resulted graph remains the same.

Lemma 3. *Let G be any given graph, and $\mathcal{V} = \{V_1, V_2, ..., V_m\}$ is any sub-automorphism partition of G. Suppose $Ocp_1(G, \mathcal{V}, V_{i_1}), ..., Ocp_N(G, \mathcal{V}, V_{i_N})$ is any orbit copying operation sequence performed on G, with $N \geq 1$ and $i_n \in \{1, 2, ..., m\}$ where $1 \leq n \leq N$ and let the resulted graph be G_N. If we interchange the order of any two successive operations (which is called as an* operation transposition*), without loss of generality, say, $Ocp_j(G, \mathcal{V}, V_{i_j})$ and $Ocp_{j+1}(G, \mathcal{V}, V_{i_{j+1}})$, where $1 \leq j \leq N - 1$ and let the so resulted graph be G'_N, then G_N and G'_N are actually the same graph, or equivalently speaking, $G_N \cong G'_N$.*

Proof. Let the graph after the n-th operation be G_n and G'_n under the two operation sequences, respectively, and let the vertex partition after the n-th operation be $\mathcal{V}^{(n)}$ and $(\mathcal{V}^{(n)})'$, respectively, where $1 \leq n \leq N$. To prove the lemma, we need only to prove that $G_{j+1} \cong G'_{j+1}$. There are two cases to be considered:

(1) $i_j = i_{j+1}$, then $Ocp_j(G, \mathcal{V}, V_{i_j})$ and $Ocp_{j+1}(G, \mathcal{V}, V_{i_{j+1}})$ are in fact operated on the same cell. Therefore it is trivial to see that G_{j+1} and G'_{j+1} must be the same.

(2) $i_j \neq i_{j+1}$, but $CAdjM(\mathcal{V}^{(j-1)})[i_j, i_{j+1}] = 0$. Then $Ocp_j(G, \mathcal{V}, V_{i_j})$ and $Ocp_{j+1}(G, \mathcal{V}, V_{i_{j+1}})$ are actually independent. More specifically, let $V_{i_j} = \{v_1, ..., v_s\}$, $V_{i_{j+1}} = \{u_1, ..., u_t\}$, and let $\mathcal{V}^{(j-1)} = (\mathcal{V}^{(j-1)})' = \{V_1^{(j-1)}, ..., V_m^{(j-1)}\}$. We suppose $V_{i_j}^{(j-1)} = \{v_1, ..., v_s,\ v_1^{(1)}, ..., v_s^{(1)}, ..., v_1^{(p-1)}, ..., v_s^{(p-1)}\}$, and $V_{i_{j+1}}^{(j-1)} = \{u_1, ..., u_t, u_1^{(1)}, ..., u_t^{(1)}, ..., u_1^{(q-1)}, ..., u_s^{(q-1)}\}$, where $p, q \geq 1$ are positive integers. After the two operations in the order $Ocp_j(G, \mathcal{V}, V_{i_j})$ and $Ocp_{j+1}(G, \mathcal{V}, V_{i_{j+1}})$, we suppose $V_{i_j}^{(j+1)} = \{v_1, ..., v_s, v_1^{(1)}, ..., v_s^{(1)}, ..., v_1^{(p)}, ..., v_s^{(p)}\}$, and $V_{i_{j+1}}^{(j+1)} = \{u_1, ..., u_t, u_1^{(1)}, ..., u_t^{(1)}, ..., u_1^{(q)}, ..., u_t^{(q)}\}$. And if we exchange the order of two operations, then we suppose $V_{i_j}^{(j+1)} = \{v_1, ..., v_s, v_1^{(1)}, ..., v_s^{(1)}, ..., (v_1^{(p)})', ..., (v_s^{(p)})'\}$, and $V_{i_{j+1}}^{(j+1)} = \{u_1, ..., u_t, u_1^{(1)}, ..., u_t^{(1)}, ..., (u_1^{(q)})', ..., (u_t^{(q)})'\}$. We now construct a mapping from $V(G_{j+1})$ to $V(G'_{j+1})$ such that $f(v_k^{(p)}) = (v_k^{(p)})'$ and $f(u_l^{(q)}) = (u_l^{(q)})'$ where $1 \leq k \leq s$ and $1 \leq l \leq t$, and $f(v) = v$ for any $v \in V(G_{j-1}) = V(G'_{j-1})$. We next show that f is an isomorphism mapping between G_{j+1} and G'_{j+1}. First, it's trivial to see that f is a bijection. Next, to prove that f is an isomorphism mapping, then we should show that for each $e = (u, v) \in E(G_{j+1})$, $(f(u), f(v)) \in E(G'_{j+1})$. Then there are four different types of e which we now handle one by one.

(a) Both u and v in $V(G_{j-1})$. Then $(f(u), f(v)) = (u, v) \in E(G_{j-1})$, so $(f(u), f(v)) \in E(G'_{j+1})$, since orbit copying operations will not remove any edges from the graph.

(b) $u \in V(G_{j-1})$ and $v \in \{v_1^{(p)}, ..., v_s^{(p)}\}$(or $u \in V(G_{j-1})$ and $v \in \{u_1^{(q)}, ..., u_t^{(q)}\}$). Then v is added by $Ocp_j(G, \mathcal{V}, V_{i_j})$. Without loss of generality, let $v = v_k^{(p)}$, then $(f(u), f(v)) = (u, (v_k^{(p)})')$. Since we add (u, v) through $Ocp_j(G, \mathcal{V}, V_{i_j})$ when the order of the two operations is $Ocp_j(G, \mathcal{V}, V_{i_j})$, $Ocp_{j+1}(G, \mathcal{V}, V_{i_{j+1}})$ because $(u, v_k) \in E(G_{j-1})$, $(u, (v_k^{(p)})')$ must also be added when $Ocp_j(G, \mathcal{V}, V_{i_j})$ is performed after we exchange the order of the two operations.

(c) Both u and v in $\{v_1^{(p)}, ..., v_s^{(p)}\}$(or both u and v in $\{u_1^{(q)}, ..., u_t^{(q)}\}$). Let $u = v_l^{(p)}$ and $v = v_k^{(p)}$, then $(f(u), f(v)) = ((v_l^{(p)})', (v_k^{(p)})')$. Since we add (u, v) through $Ocp_j(G, \mathcal{V}, V_{i_j})$ when the order of the two operations is $Ocp_j(G, \mathcal{V}, V_{i_j})$, $Ocp_{j+1}(G, \mathcal{V}, V_{i_{j+1}})$ because $(v_l, v_k) \in E(G_{j-1})$ (in fact, because $(v_l, v_k) \in E(G)$, $((v_l^{(p)})', (v_k^{(p)})')$ must also be added when $Ocp_j(G, \mathcal{V}, V_{i_j})$ is performed after we exchange the order of the two operations.

(d) $u \in \{v_1^{(p)}, ..., v_s^{(p)}\}$ and $v \in \{u_1^{(q)}, ..., u_t^{(q)}\}$ (or $u \in \{u_1^{(q)}, ..., u_t^{(q)}\}$ and $v \in \{v_1^{(p)}, ..., v_s^{(p)}\}$). Let $u = v_l^{(p)}$ and $v = u_k^{(q)}$, then $(f(u), f(v)) = ((v_l^{(p)})', (u_k^{(q)})')$. Since u is added through $Ocp_j(G, \mathcal{V}, V_{i_j})$ and v is added through $Ocp_{j+1}(G, \mathcal{V}, V_{i_{j+1}})$ when the order of the operations is $Ocp_j(G, \mathcal{V}, V_{i_j})$, $Ocp_{j+1}(G, \mathcal{V}, V_{i_{j+1}})$, (u, v) is created must due to the fact that $(v_l, u_k) \in E(G_{j-1})$ (actually, due to the fact that $(v_l, u_k) \in E(G)$). Therefore, after we exchange the order of the two operations, we now add $(u_k^{(q)})'$ first in the operation $Ocp_{j+1}(G, \mathcal{V}, V_{i_{j+1}})$, and add the edge $((u_k^{(q)})', v_l)$ (also $((u_k^{(q)})', v_1^{(1)})$, ..., $((u_k^{(q)})', v_1^{(p-1)})$) simultaneously (sine $(u_k^{(q)})'$ is copied from u_k). Then, $(v_l^{(p)})'$ is added in the operation $Ocp_j(G, \mathcal{V}, V_{i_j})$, and the edge $((v_l^{(p)})', (u_k^{(q)})')$ will be added simultaneously because $(v_l^{(p)})'$ is copied from v_l and now $((u_k^{(q)})', v_l)$ has existed in the graph. This completes the whole proof of Lemma 3.

Lemma 4. *Let G be any given graph, and $\mathcal{V} = \{V_1, V_2, ..., V_m\}$ is any sub-automorphism partition of G. Suppose $Ocps = Ocp_1(G, \mathcal{V}, V_{i_1}), ..., Ocp_N(G, \mathcal{V}, V_{i_N})$ is any orbit copying operation sequence performed on G, with $N \geq 1$ and $i_n \in \{1, 2, ..., m\}$ where $1 \leq n \leq N$. Now let π be any permutation on $\{1, 2, ..., N\}$, then the operation sequence $\pi(Ocps) = Ocp_{\pi(1)}(G, \mathcal{V}, V_{i_{\pi(1)}}), ..., Ocp_{\pi}(N)(G, \mathcal{V}, V_{i_{\pi(N)}})$ on G produces the same resulted graph as $Ocps$.*

Proof. Due to the well known result in basic permutation group theory that any permutation could be represented as a composition of a series of transpositions, we can then obtain the operation sequence $\pi(Ocps)$ through a series of operation transpositions on $Ocps$. Since each operation transposition will not affect the graph resulted, by Lemma 3, we conclude that $\pi(Ocps)$ will produce the same graph as $Ocps$.

Now it's ready for us to give the fundamental theorem related to the orbit copying operation, which is just a generalization of Lemma 1 and 2.

Theorem 1. *Let G be any given graph, and $\mathcal{V} = \{V_1, V_2, ..., V_m\}$ is any sub-automorphism partition of G. Suppose $Ocps = Ocp_1(G, \mathcal{V}, V_{i_1}), ..., Ocp_N(G, \mathcal{V}, V_{i_N})$ is any orbit copying operation sequence performed on G, with $N \geq 1$ and $i_n \in \{1, 2, ..., m\}$ where $1 \leq n \leq N$. Let the resulted vertex partition and the corresponding graph be $\mathcal{V}^{(N)}$ and G_N, respectively. Then $\mathcal{V}^{(N)}$ is a sub-automorphism partition of G_N.*

Proof. Suppose in the indices i_1, i_2, ..., i_N, 1 occurs k_1 times, 2 occurs k_2 times, ..., and m occurs k_m times, where each $k_j \geq 0$, for $1 \leq j \leq m$. Then according to Lemma 4, if we first perform $Ocp(G, \mathcal{V}, V_1)$ k_1 times, then perform $Ocp(G, \mathcal{V}, V_2)$ k_2 times, and so on until we finally perform $Ocp(G, \mathcal{V}, V_m)$ k_m times, the so resulted graph will also be G_N. After we first perform $Ocp(G, \mathcal{V}, V_1)$ k_1 times, the resulted partition $\mathcal{V}^{(k_1)}$ is a sub-automorphism partition of G_{k_1}, according to Lemma 2. But notice that, now, the hypothesis of Lemma 2 holds on G_{k_1} and $\mathcal{V}^{(k_1)} = \{V_1^{(k_1)}, V_2^{(k_1)}, ..., V_m^{(k_1)}\} = \{V_1^{(k_1)}, V_2, ..., V_m\}$. So after we perform $Ocp(G_{k_1}, \mathcal{V}^{(k_1)}, V_2)$ k_2 times, the resulted partition $\mathcal{V}^{(k_1+k_2)}$ is again a sub-automorphism partition of $G_{k_1+k_2}$, also according to Lemma 2. This process then could be repeated until we perform $Ocp(G, \mathcal{V}, V_m)$ k_m times and obtain the partition $\mathcal{V}^{(\sum_{j=1}^{m} k_j)} = \mathcal{V}^{(N)}$, which is a sub-automorphism partition of $G_{\sum_{j=1}^{m} k_j} = G_N$.

3.3 The Anonymization Procedure

Given the network G that needs to be anonymized before releasing and its automorphism partition $Orb(G) = \{V_1, V_2, ..., V_m\}$, we now propose the following anonymization procedure (as shown in Algorithm 1) to produce a graph G' which is k-symmetric. The idea is simply repeating the orbit copying operation on $V_i \in Orb(G)$ if $|V_i| \leq k$, for $1 \leq i \leq m$.

We next prove that the graph G' produced by Algorithm 1 is k-symmetric and unique, as claimed in Theorem 2 and 3. For uniqueness of G', we mean that if we change the processing order of the cells in the anonymization procedure, the resulted graph remains the same as G'.

Algorithm 1: Anonymization

```
Input: the network $G$ to be anonymized and its automorphism partition $Orb(G) = \{V_1, V_2, ..., V_m\}$; the
       specified threshold $k$
Output: an anonymized graph $G'$ w.r.t $G$, which is $k$-symmetric
for $1 \le i \le m$ do
    if $|V_i| \ge k$ then
        Continue;
    end
    else
        Let $V_i' = V_i$;
        while $|V_i'| < k$ do
            $Ocp(G, Orb(G), V_i)$;
            $V_i' = V_i' \cup V_i$;
        end
    end
end
```

Theorem 2. *The graph G' produced by the anonymization procedure is k-symmetric.*

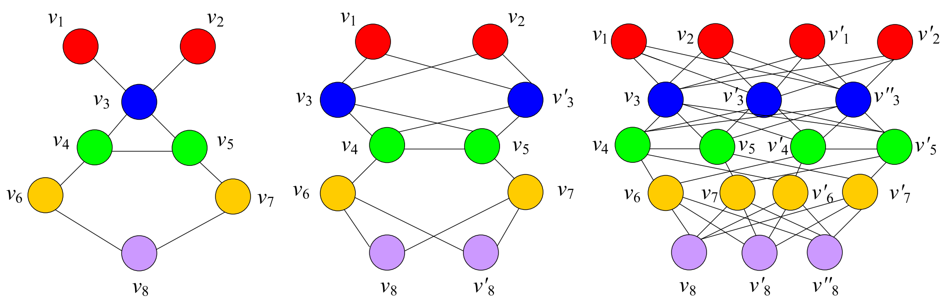
Proof. The anonymization procedure could actually be treated as a series of orbit copying operations. Suppose the resulted partition is $\mathcal{V}'$, then according to Theorem 1, $\mathcal{V}'$ is a sub-automorphism partition of G', and therefore $\mathcal{V}' \preccurlyeq Orb(G')$. Since $\forall V \in \mathcal{V}'$, $|V| \ge k$, then $\forall U \in Orb(G')$, $|U| \ge k$ either. Hence G' is k-symmetric.

Theorem 3. *The graph G' produced by the anonymization procedure is unique. More specifically, let π be any permutation of $\{1, 2, ..., m\}$. If in the anonymization procedure, we process cells in the order $V_{\pi(1)}, V_{\pi(2)}, ..., V_{\pi(m)}$ instead of $V_1, V_2, ..., V_m$, then the resulted graph G_π is the same as G', or equivalently speaking, $G_\pi \cong G'$.*

Proof. Similarly as in the proof of Theorem 2, we could actually treat the anonymization procedure as a series of orbit copying operations. Without loss of generalization, suppose V_1 is copied k_1 times, V_2 is copied k_2 times, ..., and V_m is copied k_m times in the anonymization procedure, where each $k_j \ge 0$, for $1 \le j \le m$. Let $N = \sum_{j=1}^{m} k_j$, then we could represent the orbit copying operations as a sequence: $Ocps = Ocp_1(G, Orb(G), V_1)$, ..., $Ocp_{k_1}(G, Orb(G), V_1)$, $Ocp_{k_1+1}(G, Orb(G), V_2)$, ..., $Ocp_{k_1+k_2}(G, Orb(G), V_2)$, ..., $Ocp_{\sum_{j=1}^{m-1} k_j+1}(G, Orb(G), V_m)$, ..., $Ocp_{\sum_{j=1}^{m-1} k_j+k_m}(G, Orb(G), V_m)$. Now suppose the processing order becomes $V_{\pi(1)}, V_{\pi(2)}, ..., V_{\pi(m)}$. According to the anonymization procedure, the number of operations $Ocp(G, Orb(G), V_{\pi(i)})$ remains unchanged, for each $V_{\pi(i)}$, and therefore the new operation sequence $Ocps_\pi = Ocp_1(G, Orb(G), V_{\pi(1)})$, ..., $Ocp_{k_{\pi(1)}}(G, Orb(G), V_{\pi(1)})$, ..., $Ocp_{\sum_{j=1}^{m-1} k_{\pi(j)}+1}(G, Orb(G), V_{\pi(m)})$, ..., $Ocp_{\sum_{j=1}^{m-1} k_{\pi(j)}+k_{\pi(m)}}(G, Orb(G), V_{\pi(m)})$ is just a rearrangement of the operations in $Ocps$. Hence according to Lemma 4, the resulted graph G_π after $Ocps_\pi$ is the same as the resulted graph G' after $Ocps$, namely, $G_\pi \cong G'$.

Theorem 3 is a nice result. Given a graph G, we could use different methods to find its automorphism partition $Orb(G)$. But in general, different methods may result $Orb(G)$ with different orbit orders, and it's hard for us to enforce a consistent ordering since we even don't know how many different automorphism partition generating algorithms existing at present. Theorem 3 solves this problem and make our anonymization procedure independent of the underlying automorphism partition generator. Therefore, given G and $Orb(G)$, the resulted graph G' is determined uniquely.

Example 3. As shown in Fig 4(a), we have $Orb(G) = \{W_1, W_2, W_3, W_4, W_5\}$, where $W_1 = \{v_1, v_2\}, W_2 = \{v_3\}, W_3 = \{v_4, v_5\}, W_4 = \{v_6, v_7\}, W_5 = \{v_8\}$. Suppose now $k = 2$, then W_2 and W_5 need to be copied if we want to produce a 2-symmetric graph based on G. Fig 4(b) shows the graph G' after the anonymization procedure. Now we get a vertex partition $\mathcal{V}' = V_1, V_2, V_3, V_4, V_5$ of $V(G')$, where $V_1 = \{v_1, v_2\}, V_2 = \{v_3, v_3'\}, V_3 = \{v_4, v_5\}, V_4 = \{v_6, v_7\}, V_5 = \{v_8, v_8'\}$. Each cell of $\mathcal{V}'$ contains at least 2 vertices that are structurally equivalent and we can easily check that $\mathcal{V}$ is a sub-automorphism partition of G'. Actually, in this case $Orb(G') = \mathcal{V}'$, and thus G' is 2-symmetric. Fig 4(c) shows the graph G' after the anonymization procedure when $k = 3$. Here, since none of the 5 orbits of $Orb(G)$ satisfies the k-symmetry constraint, all of them should be copied in the anonymization procedure.



(a) The original graph G (b) The anonymized graph G' when $k = 2$ (c) The anonymized graph G' when $k = 3$

Fig. 4. Illustration of the anonymization procedure

The time complexity of the anonymization procedure is polynomial if the time required by computing the automorphism partition of the original graph G is not considered. Specifically, suppose the number of cells in $\mathcal{V}$ which contain less than k vertices is N, and let these cells to be $V_{i_1}, ..., V_{i_N}$. Suppose the number of orbit copying operation performed is $k_1, ...k_N$, respectively. Then the total number of vertices introduced is $\sum_{j=1}^{N} k_j |V_{i_j}| \leq (k-1)|V(G)|$, since each $k_j \leq k - 1$ and $\sum_{j=1}^{N} |V_{i_j}| \leq |V(G)|$. And the total number of edges introduced is less than $\sum_{j=1}^{N} k_j |V_{i_j}|(k|V(G)|) \leq k(k-1)|V(G)|^2$. Since usually k is much less than $|V(G)|$, we then have that the worst case running time of the anonymiztion procedure is $O(|V(G)|^2)$. However, computing the automorphism partition of G is not trivial and we shall give some remarks on this problem later in this paper.

3.4 Utility

A critical problem that needs to be considered in any privacy protection model is the *utility* of the published data. So do our k-symmetry model. Since new vertices and edges would be introduced during the anonymization procedure, the resulted graph G' is unlikely to share similar statistical properties as the original graph G. We consider the utility problem in this section.

Orbit Linkage Pattern Recall the anonymization procedure, and note that besides the property that the resulted graph is k-symmetric, it also preserves the *linkage pattern* between different orbits of the original graph.

We argue here that the properties of graph is highly influenced by this linkage pattern. Although we could not give theoretical analysis at the current time, the following experimental example strongly supports this argument.

Example 4 (An experiment). Fig 5 shows two link patterns between two orbits each containing 4 vertices. Each vertex has degree 2.

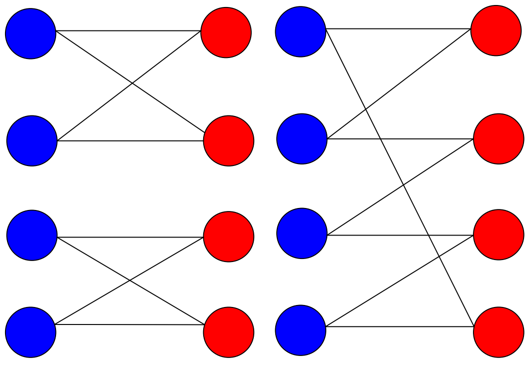
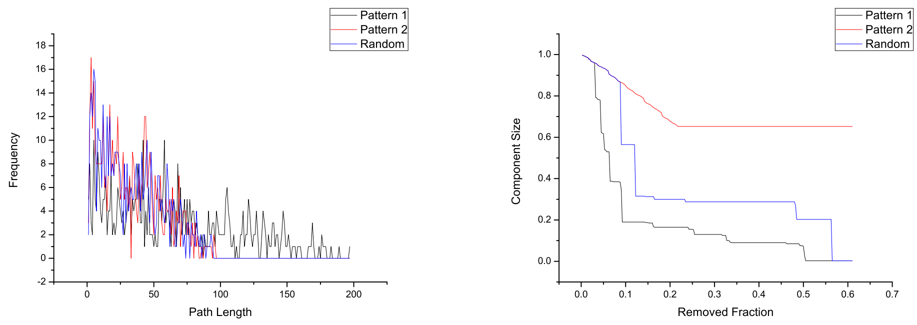

(a) Link Pattern 1 (b) Link Pattern 2

Fig. 5. Link patterns used in Example 4

We now construct graphs having exactly 100 such orbits (namely, 400 vertices). Suppose O_i denotes the i-th orbit, $1 \leq i \leq 100$ and the edges of the following mentioned graphs are only between O_j and O_{j+1}, for $1 \leq j \leq 99$. We construct following three type of graphs:

(1) G_1 is a graph such that O_1 and O_2 are linked by Link Pattern 2 (just for G_1 to be connected), and O_j and O_{j+1} are linked by Link Pattern 1, for $2 \leq j \leq 99$.
(2) G_2 is a graph such that O_j and O_{j+1} are linked by Link Pattern 2, for $1 \leq j \leq 99$.
(3) G_s is a graph such that O_j and O_{j+1} are linked by Link Pattern 1 or 2, with equal probability, for $1 \leq j \leq 99$.



(a) Shortest Path Length Distribution (b) Network Resilience

Fig. 6. Network properties for graphs generated in Example 4

Fig 6 shows the experimental results of some network properties of the three graphs generated above. Two property measures are used here. One is *Shortest Path Length Distribution*, which is calculated by randomly sampling 500 vertex pairs from the graph and computing the length of the shortest path between them. The other is *Network Resilience*, which is to compute the relative size of the largest connected component when vertices of the graph are removed in a decreasing order of their degrees.

We can see from the result that the properties of graphs are quite different, especially for G_1 and G_2, which supports our argument that the properties of graph are highly influenced by the linkage pattern between different orbits.

Graph Backbone The concept of orbit linkage pattern described in the last part so far is intuitive and not theoretically strict, which makes it hard to go any further based on just this intuition. In this part, we shall give deeper insight into the notion of orbit linkage pattern by generalizing it into what we call as *graph backbone*.

Definition 4 (Graph Generalization and Reduction).

Let G be a given graph, $\mathcal{V} = \{V_1, V_2, ..., V_m\}$ be a sub-automorphism partition of G.

Suppose $Ocps = Ocp_1(G, \mathcal{V}, V_{i_1}), ..., Ocp_N(G, \mathcal{V}, V_{i_N})$ be an orbit copying operation sequence on G with respect to $\mathcal{V}$, where $N \geq 1$ is a positive integer and $i_n \in \{1, 2, ..., m\}$ for $1 \leq n \leq N$. Then the resulted graph after $Ocps$ is called a ***generalization*** *of G, with respect to $\mathcal{V}$ and $Ocps$, and we denote it as $G(\mathcal{V}, Ocps)$.*

Conversely, if there exists a subgraph H of G, a sub-automorphism partition $\mathcal{V}_H$ of H, and an orbit copying operation sequence $Ocps_H$ on H, with respect to $\mathcal{V}^H$, such that $G \cong H(\mathcal{V}^H, Ocps^H)$, then H is called a ***reduction*** *of G, with respect to $\mathcal{V}$, and we denote H as $G^{-1}(\mathcal{V}, Ocps^H)$.*

In particular, G is both a generalization and reduction of itself, with respect to $\mathcal{V}$ and an empty orbit copying operation sequence.

In terms of Definition 4, the resulted graph G' of the anonymization procedure is a generalization of the input graph G, with respect to $Orb(G)$ and the corresponding orbit copying operation sequence $Ocps$, namely, $G' = G(Orb(G), Ocps)$.

We now focus on the discussion of graph reduction since it is more related to the concept of graph backbone, which we will define shortly. For convenience, we introduce the following notation $\leqq$: If H is a reduction of G, with respect to $\mathcal{V}$, and $\mathcal{V}^H$ is the corresponding sub-automorphism partition of H, as in Definition 4, then $(H, \mathcal{V}^H) \leqq (G, \mathcal{V})$. Furthermore, if $(H, \mathcal{V}^H) \leqq (G, \mathcal{V})$, then there is a vector $\mathbf{k} = (k_1, k_2, ..., k_m)$ such that $k_i \geq 0$ is the number of operation $Ocp(H, \mathcal{V}^H, V_i^H)$ performed. We call $\mathbf{k}$ as the *orbit copying frequency vector*, or briefly, the *ocf-vector*. Since the graph after each orbit copying operation is uniquely determined, and rearranging the order of the operations will not affect the result, the vector $\mathbf{k}$ is then unique if G, $\mathcal{V}$, H and $\mathcal{V}^H$ are given. We hence use the notation $\mathbf{k}^{H,\mathcal{V}^H,G,\mathcal{V}}$ to emphasize this uniqueness relation. In the context where $\mathcal{V}$ and $\mathcal{V}^H$ are clear, we also use the notation $\mathbf{k}^{H,G}$, for abbreviation. Furthermore, we use the notation $(H, \mathcal{V}^H, \mathbf{k}^{H,G}) \rightarrow (G, \mathcal{V})$ to emphasize the fact that $(G, \mathcal{V})$ could be obtained from $(H, \mathcal{V}^H)$ by applying corresponding number of orbit copying operations according to $\mathbf{k}^{H,G}$.

First, it's clear that a graph G may have multiple reductions, with respect to the given sub-automorphism partition $\mathcal{V}$. Let $\mathcal{R}(G, \mathcal{V})$ be the set containing all the reductions of G with respect to $\mathcal{V}$, namely, $\mathcal{R}(G, \mathcal{V}) = \{H | (H, \mathcal{V}^H) \leqq (G, \mathcal{V})\}$. Since $G \in \mathcal{R}(G, \mathcal{V})$, $\mathcal{R}(G, \mathcal{V})$ is not empty, we can then define a relation $\leq$ on $\mathcal{R}(G, \mathcal{V})$ such that $H_1 \leq H_2$ if and only if $(H_1, \mathcal{V}^{H_1}) \leqq (H_2, \mathcal{V}^{H_2})$, for any H_1 and H_2 in $\mathcal{R}(G, \mathcal{V})$. It's trivial to check that $\leq$ is reflexive, asymmetric, and transitive and hence it is a partial order.

Suppose H_1 and H_2 are two elements in $\mathcal{R}(G, \mathcal{V})$. If there is an $H_u \in \mathcal{R}(G, \mathcal{V})$ such that $H_1 \leq H_u$ and $H_2 \leq H_u$, then H_u is called an *upper bound* of H_1 and H_2. Similarly, if there is an $H_l \in \mathcal{R}(G, \mathcal{V})$ such that $H_l \leq H_1$ and $H_l \leq H_2$, then H_l is called a *lower bound* of H_1 and H_2. Our next result is that for any two H_1 and H_2 are two elements in $\mathcal{R}(G, \mathcal{V})$, there is a *least upper bound* and a *greatest lower bound* of H_1 and H_2 in $\mathcal{R}(G, \mathcal{V})$, with respect to $\leq$. Therefore, the poset $(\mathcal{R}(G, \mathcal{V}); \leq)$ is indeed a *lattice*. But before we show this in Theorem 4, we shall give some preliminaries first.

Lemma 5. *Let A be a finite set, $|A| = n$ and R be a binary equivalence relation defined on A. Suppose $p|n$, $q|n$, $p \neq q$ and $n = p \cdot s = q \cdot t$. Assume that the following two conditions hold:*

1. *The elements in A could be partitioned into s disjoint subsets A_i, such that $\forall a \in A_i, b \in A_i$, we have $(a, b) \in R$, where $|A_i| = p$ for each $1 \leq i \leq s$.*
2. *The elements in A could be partitioned into t disjoint subsets B_j, such that $\forall a \in B_j, b \in B_j$, we have $(a, b) \in R$, where $|B_j| = q$ for each $1 \leq j \leq t$.*

Then the elements in A could be partitioned into $gcd(s, t)$ disjoint subsets C_k, such that $\forall a \in C_k, b \in C_k$, we have $(a, b) \in R$, where $|C_k| = lcm(p, q)$ for each $1 \leq k \leq gcd(s, t)$. Here $gcd(x, y)$ and $lcm(x, y)$ means the greatest common divisor *and* least common multiple *of x and y, respectively.*

Proof. Suppose A/R is the quotient set induced by R on A. Let $E \in A/R$, then $|E| = m \cdot lcm(p, q)$ where $m \geq 1$ is a positive integer. In other words, the size of every equivalence class must be a multiple of $lcm(p, q)$. We show this by contradiction. Suppose this is not the case, then there is an $F \in A/R$, $|F| = M$ such that $lcm(p, q) \nmid M$. Thus either $p \nmid M$ or $q \nmid M$. Without loss of generality, suppose $p \nmid M$, then $M = k \cdot p + r$, where $k \geq 1$ is some non-negative integer and $1 \leq r \leq M - 1$. Pick $a_1 \in F$. According to the first condition, since A_is forms a partition of A, $a_1 \in A_{i_1}$ for some $1 \leq i_1 \leq s$. But now every y in A_{i_1} must also be in F, since $(a_1, y) \in R$. So we have $A_{i_1} \subset F$. Let $F_1 = F \setminus A_{i_1}$. Next we pick $a_2 \in F_1$, and similarly $a_2 \in A_{i_2}$ for some $i_2 \neq i_1$ and hence$A_{i_2} \subset F_1$. Let $F_2 = F_1 \setminus A_{i_2} = F \setminus (A_{i_1} \cup A_{i_2})$. We can then pick $a_3 \in F_2$ and repeat the previous process. Since $M = k \cdot p + r$, this process could repeat k times and we have $|F_k| = r$, where $F_k = F \setminus (\cup_{j=1}^{k} A_{i_j})$. But if we now choose $a_{i_{k+1}} \in F_k$, the above argument shows that $a_{i_{k+1}} \in A_{i_{k+1}}$, where $i_{k+1} \neq i_j$ for $1 \leq j \leq k$. Hence $A_{i_{k+1}} \in F_k$ and therefore $|F_k| \geq |A_{i_{k+1}}| = p > r$, contradict to $|F_k| = r$. The case $q \nmid M$ could be showed similarly. Thus, M must be a multiple of $lcm(p, q)$, namely $lcm(p, q)|M$. Since F is arbitrarily chosen, we conclude that each equivalence class $E \in A/R$ must have size which is a multiple of $lcm(p, q)$. Since A/R is also a partition of A, we can then obtain the partition $\{C_k\}$ by partitioning each $E \in A/R$ into disjoint subsets each with size $lcm(p, q)$. Since $n = p \cdot s = q \cdot t$, n is a common multiple of p and q and thus $lcm(p, q)|n$. Suppose $n = lcm(p, q) \cdot l$, then l is the number of C_ks. Let $lcm(p, q) = l_1 \cdot p = l_2 \cdot q$. Here $gcd(l_1, l_2) = 1$. Since $n = p \cdot s = lcm(p, q) \cdot l$, we have $p \cdot s = l_1 \cdot p \cdot l$, namely $s = l_1 \cdot l$. Similarly, since $n = q \cdot t = lcm(p, q) \cdot l$, we have $q \cdot t = l_2 \cdot q \cdot l$, namely $t = l_2 \cdot l$. Therefore, $l|s$, $l|t$, then $l|gcd(s, t)$. Let l_3 be any common divisor of s and t, namely, $l_3|s$ and $l_3|t$. Suppose $s = k_s \cdot l_3$, $t = k_t \cdot l_3$. Since $p \cdot s = lcm(p, q) \cdot l$, we have $p \cdot l_3 \cdot k_s = p \cdot l_1 \cdot l$, i.e., $l_3 \cdot k_s = l_1 \cdot l$; and since $q \cdot t = lcm(p, q) \cdot l$, we have $q \cdot l_3 \cdot k_t = q \cdot l_2 \cdot l$, i.e., $l_3 \cdot k_t = l_2 \cdot l$. Therefore, $l_3 \cdot k_s \cdot l_2 = l_3 \cdot k_t \cdot l_1$, i.e.,$k_s \cdot l_2 = k_t \cdot l_1$. Since $gcd(l_1, l_2) = 1$, we must have $l_1|k_s$, $l_2|k_t$. Hence $l_3|l$ since $l_3 \cdot k_s = l_1 \cdot l$ and $l_3 \cdot k_t = l_2 \cdot l$. Because l_3 is chosen arbitrary, if we let $l_3 = gcd(s, t)$, we then have $gcd(s, t)|l$. Therefore, $l = gcd(c, d)$, which completes the proof of the lemma.

Lemma 6. *Suppose H_1 and H_2 are any two elements in $\mathcal{R}(G,\mathcal{V})$. Then there is a graph $H \in \mathcal{R}(G,\mathcal{V})$ such that $H \leq H_1$, $H \leq H_2$, and what's more, for any other $H' \in \mathcal{R}(G,\mathcal{V})$ such that $H' \leq H_1$ and $H' \leq H_2$, it holds that $H' \leq H$. We denote $H = H_1 \wedge H_2$. In other words, any two elements in $\mathcal{R}(G,\mathcal{V})$ has a greatest lower bound in $\mathcal{R}(G,\mathcal{V})$.*

Proof. Let $\mathcal{V}^{H_1}$ and $\mathcal{V}^{H_2}$ be the corresponding sub-automorphism partition of H_1 and H_2, respectively. Let $\mathbf{k}^{H_1,G}$ and $\mathbf{k}^{H_2,G}$ be the corresponding ocf-vector. Then $(H_1,\mathcal{V}^{H_1},\mathbf{k}^{H_1,G}) \to (G,\mathcal{V})$ and $(H_2,\mathcal{V}^{H_2},\mathbf{k}^{H_2,G}) \to (G,\mathcal{V})$. We define a binary relation R_i on each $V_i \in \mathcal{V}$ such that $\forall u,v \in V_i$, $(u,v) \in R_i$ if and only if $u = v$ or u and v could be copied to each other. Then it is easy to check that R_i is an equivalence relation. Since V_i could be obtained by applying $Ocp(H_1,\mathcal{V}^{H_1},V_i^{H_1})$ $k_i^{H_1,G}$ times, the vertices in V_i could be partitioned into $|V_i^{H_1}|$ disjoint subsets with each containing $k_i^{H_1,G}$ mutually equivalent vertices according to R_i. On the other hand, since V_i could also be obtained by applying $Ocp(H_2,\mathcal{V}^{H_2},V_i^{H_2})$ $k_i^{H_2,G}$ times, the vertices in V_i could be partitioned into $|V_i^{H_2}|$ disjoint subsets with each containing $k_i^{H_2,G}$ mutually equivalent vertices according to R_i. Then according to Lemma 5, the vertices in V_i could then be partitioned into $gcd(|V_i^{H_1}|,|V_i^{H_2}|)$ disjoint subsets with each containing $lcm(k_i^{H_1,G},k_i^{H_2,G})$ mutually equivalent vertices according to R_i. We then form a vertex partition $\mathcal{V}^H$ by picking one vertex from each of these $gcd(|V_i^{H_1}|,|V_i^{H_2}|)$ subsets to consist the cell V_i^H, for each V_i, and refer the corresponding graph to be H. It is easy to show that $\mathcal{V}^H$ is a sub-automorphism partition of H, by using a similar trick as in the proof of Lemma 1. Define two vectors $\mathbf{k}^{H,H_1}$ and $\mathbf{k}^{H,H_2}$ such that $k_i^{H,H_1} = |V_i^{H_1}|/gcd(|V_i^{H_1}|,|V_i^{H_2}|)$ and $k_i^{H,H_2} = |V_i^{H_2}|/gcd(|V_i^{H_1}|,|V_i^{H_2}|)$. Then $(H,\mathcal{V}^H,\mathbf{k}^{H,H_1}) \to (H_1,\mathcal{V}^{H_1})$ and $(H,\mathcal{V}^H,\mathbf{k}^{H,H_2}) \to (H_2,\mathcal{V}^{H_2})$. Hence $H \leq H_1$ and $H \leq H_2$. Define vector $\mathbf{k}^{H,G}$ to be $k_i^{H,G} = k_i^{H,H_1} \cdot k_i^{H_1,G} = k_i^{H,H_2} \cdot k_i^{H_2,G} = |V_i|/gcd(|V_i^{H_1}|,|V_i^{H_2}|)$. Then $(H,\mathcal{V}^H,\mathbf{k}^{H,G}) \to (G,\mathcal{V})$ and thus $H \in \mathcal{R}(G,\mathcal{V})$. Now suppose $H' \in \mathcal{R}(G,\mathcal{V})$ be any element that $H' \leq H_1$ and $H' \leq H_2$. Then similarly we have $(H',\mathcal{V}^{H'},\mathbf{k}^{H',H_1}) \to (H_1,\mathcal{V}^{H_1})$ and $(H',\mathcal{V}^{H'},\mathbf{k}^{H',H_2}) \to (H_2,\mathcal{V}^{H_2})$. Since $H' \in \mathcal{R}(G,\mathcal{V})$, we also have $(H',\mathcal{V}^{H'},\mathbf{k}^{H',G}) \to (G,\mathcal{V})$ and therefore $k_i^{H',G} = k_i^{H',H_1} \cdot k_i^{H_1,G} = k_i^{H',H_2} \cdot k_i^{H_2,G} = |V_i|/|V_i^{H'}|$. Since $k_i^{H',H_1} = |V_i^{H_1}|/|V_i^{H'}|$ and $k_i^{H',H_2} = |V_i^{H_2}|/|V_i^{H'}|$, we have $(|V_i^{H'}|) \mid gcd(|V_i^{H_1}|,|V_i^{H_2}|)$. And since $|V_i| = k_i^{H,G} \cdot gcd(|V_i^{H_1}|,|V_i^{H_2}|) = k_i^{H',G} \cdot |V_i^{H'}|$, then $k_i^{H,G} \mid k_i^{H',G}$. Define vector $\mathbf{k}^{H',H}$ to be $k_i^{H',H} = k_i^{H',G}/k_i^{H,G}$. Then $(H',\mathcal{V}^{H'},\mathbf{k}^{H',H}) \to (H,\mathcal{V}^H)$. Therefore, $H' \leq H$ and this completes the whole proof of the lemma.

Lemma 7. *Suppose H_1 and H_2 are any two elements in $\mathcal{R}(G,\mathcal{V})$. Then there is a graph $H \in \mathcal{R}(G,\mathcal{V})$ such that $H_1 \leq H$, $H_2 \leq H$, and what's more, for any other $H' \in \mathcal{R}(G,\mathcal{V})$ such that $H_1 \leq H'$ and $H_2 \leq H'$, it holds that $H \leq H'$. We denote $H = H_1 \vee H_2$. In other words, any two elements in $\mathcal{R}(G,\mathcal{V})$ has a least upper bound in $\mathcal{R}(G,\mathcal{V})$.*

Proof. Let $\mathcal{V}^{H_1}$ and $\mathcal{V}^{H_2}$ be the corresponding sub-automorphism partition of H_1 and H_2, respectively. Let $\mathbf{k}^{H_1,G}$ and $\mathbf{k}^{H_2,G}$ be the corresponding ocf-vector. Define vector $\mathbf{k}^{H,G}$ such that $k_i^{H,G} = gcd(k_i^{H_1,G},k_i^{H_2,G})$, and define vector $\mathbf{k}^{H_1,H}$ and $\mathbf{k}^{H_2,H}$ to be $k_i^{H_1,H} = k_i^{H_1,G}/k_i^{H,G}$ and $k_i^{H_2,H} = k_i^{H_2,G}/k_i^{H,G}$, respectively. Let the graph produced by applying orbit copying operations on H_1 with respect to $\mathcal{V}^{H_1}$, according to $\mathbf{k}^{H_1,H}$, be G_1, and let the corresponding sub-automorphism partition be $\mathcal{V}^{G_1}$. Similarly, let the graph produced by applying orbit copying operations on H_2 with respect to $\mathcal{V}^{H_2}$, according to $\mathbf{k}^{H_2,H}$, be G_2, and let the corresponding sub-automorphism partition be $\mathcal{V}^{G_2}$. Since $(H_1,\mathcal{V}^{H_1},\mathbf{k}^{H_1,G}) \to (G,\mathcal{V})$, then $(H_1,\mathcal{V}^{H_1},\mathbf{k}^{H_1,H}) \to (G_1,\mathcal{V}^{G_1})$, and $(G_1,\mathcal{V}^{G_1},\mathbf{k}^{H,G}) \to (G,\mathcal{V})$. Similarly, since $(H_2,\mathcal{V}^{H_2},\mathbf{k}^{H_2,G}) \to (G,\mathcal{V})$, then $(H_2,\mathcal{V}^{H_2},\mathbf{k}^{H_2,H}) \to (G_2,\mathcal{V}^{G_2})$, and $(G_2,\mathcal{V}^{G_2},\mathbf{k}^{H,G}) \to (G,\mathcal{V})$. Therefore, we must have $G_1 = G_2$ and $\mathcal{V}^{G_1} = \mathcal{V}^{G_2}$.

We then use H to denote this common graph and use $\mathcal{V}^H$ to denote the corresponding common sub-automorphism partition. It's now clear that $H_1 \leq H$ and $H_2 \leq H$, since $(H_1, \mathcal{V}^{H_1}, \mathbf{k}^{H_1,H}) \to (H, \mathcal{V}^H)$ and $(H_2, \mathcal{V}^{H_2}, \mathbf{k}^{H_2,H}) \to (H, \mathcal{V}^H)$. And it's also clear that $H \in \mathcal{R}(G, \mathcal{V})$ since now $(H, \mathcal{V}^H, \mathbf{k}^{H,G}) \to (G, \mathcal{V})$. Suppose $H' \in \mathcal{R}(G, \mathcal{V})$ is any other element such that $H_1 \leq H'$, and $H_2 \leq H'$. Then similarly we will have $(H_1, \mathcal{V}^{H_1}, \mathbf{k}^{H_1,H'}) \to (H', \mathcal{V}^{H'})$, $(H', \mathcal{V}^{H'}, \mathbf{k}^{H',G}) \to (G, \mathcal{V})$ and $(H_2, \mathcal{V}^{H_2}, \mathbf{k}^{H_2,H'}) \to (H', \mathcal{V}^{H'})$, $(H', \mathcal{V}^{H'}, \mathbf{k}^{H',G}) \to (G, \mathcal{V})$. Therefore we must have $k_i^{H',G} | k_i^{H_1,G}$ and $k_i^{H',G} | k_i^{H_2,G}$. Hence $k_i^{H',G} | gcd(k_i^{H_1,G}, k_i^{H_2,G})$, i.e., $k_i^{H',G} | k_i^{H,G}$. Since $k_i^{H_1,G} = k_i^{H_1,H} \cdot k_i^{H,G} = k_i^{H_1,H'} \cdot k_i^{H',G}$, and $k_i^{H_2,G} = k_i^{H_2,H} \cdot k_i^{H,G} = k_i^{H_2,H'} \cdot k_i^{H',G}$, we thus have $k_i^{H_1,H'}/k_i^{H_1,H} = k_i^{H_2,H'}/k_i^{H_2,H} = k_i^{H,G}/k_i^{H',G}$. Therefore, if we define vector $\mathbf{k}^{H,H'}$ to be $k_i^{H,H'} = k_i^{H,G}/k_i^{H',G}$, then we have $(H, \mathcal{V}^H, \mathbf{k}^{H,H'}) \to (H', \mathcal{V}^{H'})$, which means $H \leq H'$ and thus completes the proof of the lemma.

Theorem 4 is then a straightforward result according to Lemma 6 and 7.

Theorem 4. *The poset* $(\mathcal{R}(G, \mathcal{V}); \leq)$ *is a lattice.*

Since $(\mathcal{R}(G, \mathcal{V}); \leq)$ is a lattice, we can then deem $\vee$ and $\wedge$ as two binary algebraic operations and also denote $\mathcal{R}(G, \mathcal{V})$ as $[\mathcal{R}(G, \mathcal{V}); \vee, \wedge]$, in which $\mathcal{R}(G, \mathcal{V})$ is treated as an algebraic structure. Furthermore, since $\mathcal{R}(G, \mathcal{V})$ is finite, it is also bounded with the greatest element (or the *top*) $\vee_{H \in \mathcal{R}(G,\mathcal{V})} H$ and the least element (or the *bottom*) $\wedge_{H \in \mathcal{R}(G,\mathcal{V})} H$. Actually, $\vee_{H \in \mathcal{R}(G,\mathcal{V})} H = G$ since $G \in \mathcal{R}(G, \mathcal{V})$. The element $\wedge_{H \in \mathcal{R}(G,\mathcal{V})} H$ is then called as the *backbone* of G, with respect to $\mathcal{V}$, which is formally defined in Definition 5.

Definition 5 (Graph Backbone). *Given a graph G and a sub-automorphism partition $\mathcal{V}$ of G. The least element $\wedge_{H \in \mathcal{R}(G,\mathcal{V})} H$ in the lattice $[\mathcal{R}(G, \mathcal{V}); \vee, \wedge]$ is called the* ***backbone*** *of G, with respect to $\mathcal{V}$, and we denote it as $B_{G,\mathcal{V}}$, namely, $B_{G,\mathcal{V}} = \wedge_{H \in \mathcal{R}(G,\mathcal{V})} H$.*

Fig 7 illustrates several graphs and their backbones, with respect to the given sub-automorphism partitions.

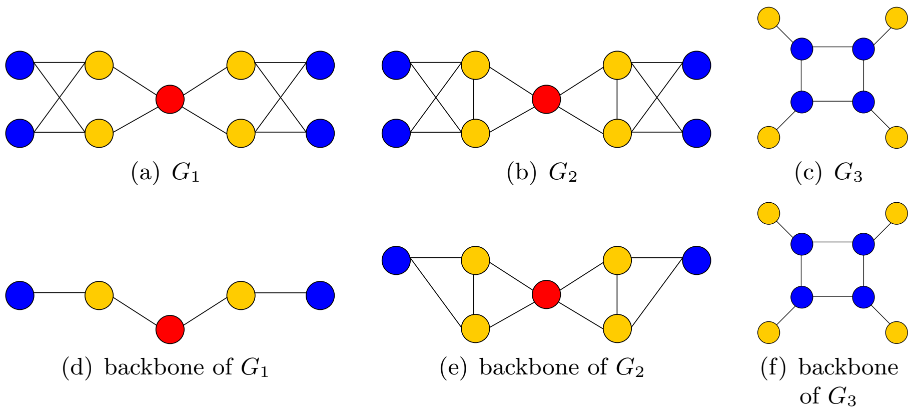


Fig. 7. Examples of graphs and their backbones

Theorem 5 states an important property of graph backbones.

Theorem 5. *Suppose G is a graph and $\mathcal{V}$ is a sub-automorphism partition of G. Let $H \in \mathcal{R}(G, \mathcal{V})$, and $\mathcal{V}^H$ be the corresponding sub-automorphism partition of H. Then $B_{G,\mathcal{V}} = B_{H,\mathcal{V}_H}$.*

Proof. Since $H \in \mathcal{R}(G, \mathcal{V})$, then $B_{G,\mathcal{V}} \leq H$ and therefore $B_{G,\mathcal{V}} \in \mathcal{R}(H, \mathcal{V}^H)$. Hence we must have $B_{H,\mathcal{V}_H} \leq B_{G,\mathcal{V}}$. But then we have $(B_{H,\mathcal{V}_H}, \mathcal{V}^{B_{H,\mathcal{V}_H}}, \mathbf{k}^{B_{H,\mathcal{V}_H},B_{G,\mathcal{V}}}) \to (B_{G,\mathcal{V}}, \mathcal{V}^{B_{G,\mathcal{V}}})$. Define vector $\mathbf{k}^{B_{H,\mathcal{V}_H},G}$ to be $k_i^{B_{H,\mathcal{V}_H},G} = k_i^{B_{H,\mathcal{V}_H},B_{G,\mathcal{V}}} \cdot k_i^{B_{G,\mathcal{V}},G}$. Then $(B_{H,\mathcal{V}_H}, \mathcal{V}^{B_{H,\mathcal{V}_H}}, \mathbf{k}^{B_{H,\mathcal{V}_H},G}) \to (G, \mathcal{V})$ and therefore $B_{H,\mathcal{V}_H} \in \mathcal{R}(G, \mathcal{V})$. So we must have $B_{G,\mathcal{V}} \leq B_{H,\mathcal{V}_H}$ and thus $B_{G,\mathcal{V}} = B_{H,\mathcal{V}_H}$.

Furthermore, since orbit copying operations only introduce new vertices and edges, the following theorem then is straightforward.

Theorem 6. *Graph generalization conserves the backbone of the original graph, with respect to the corresponding sub-automorphism partitions of the original graph and the generalized graph.*

Proof. Suppose G is a graph and $\mathcal{V}$ is a sub-automorphism partition of G. Let G' be any generalization of G, with respect to $\mathcal{V}$, and let the resulted sub-automorphism partition of G' be $\mathcal{V}'$. Then $G \in \mathcal{R}(G', \mathcal{V}')$ and thus $B_{G,\mathcal{V}} = B_{G',\mathcal{V}'}$, according to Theorem 5.

As a result of Theorem 6 , the graph G' produced by the anonymization procedure conserves the backbone of the original graph G with respect to the given sub-automorphism partition $\mathcal{V}$ of G and the resulted sub-automorphism partition $\mathcal{V}'$ of G'.

Utility Preservation By Graph Backbone-Based Sampling Method The importance of the backbone is that it conserves the *basic* orbit linkage pattern of the original graph. In other words, removing any vertices (and their induced edges) from the backbone will lose orbit linkage pattern information of the original graph, namely, we cannot then recover the original graph back through a series of orbit copying operations. As we have shown in the previous experiment, the linkage pattern is highly related to the statistical property, the backbone is then also highly related to the statistical property of the original graph. Since our anonymization procedure preserves the backbone of the original graph, in this part, we will propose a backbone-based sampling method to approximate the statistical properties of the original graph by sampling subgraphs preserving the backbone of the original graph, from the anonymized graph.

Definition 6 (Possible World). *Let G be a given graph and $\mathcal{V} = \{V_1, V_2, ..., V_m\}$ be a sub-automorphism partition of G. Suppose the corresponding sub-automorphism partition of $B_{G,\mathcal{V}}$ is $\mathcal{B} = \{B_1, B_2, ..., B_m\}$. Let $\mathcal{P}$ be a set of properties of G, then the set $PW(G, \mathcal{V}, \mathcal{P}) = \{G_{B,\mathcal{B}} | (\exists Ocps, G_{B,\mathcal{B}} = B_{G,\mathcal{V}}(\mathcal{B}, Ocps) \wedge (G_{B,\mathcal{B}} \neq G) \wedge (P(G_{B,\mathcal{B}}) = P(G), \forall P \in \mathcal{P})\}$ is called the **possible world** of G, with respect to $\mathcal{V}$ and $\mathcal{P}$. The **size** of the possible world is then naturally defined to be $|PW(G, \mathcal{V}, \mathcal{P})| + 1$ (1 for G itself).*

We now formalize our idea of backbone based sampling according to the definition of possible world.

Definition 7 (Backbone-Based Sampling). *Given the anonymized graph G', the corresponding sub-automorphism partition $\mathcal{V}'$ and the property set $\mathcal{P}$ (these are also the information released by the network publisher), a **backbone-based sampling** samples a graph from $PW(G, \mathcal{V}, \mathcal{P})$ uniformly.*

From now on in this paper, we shall just consider two property sets, namely, $\mathcal{P}_1 = \{|V(G)| = n, |E(G)| = m\}$, and $\mathcal{P}_2 = \{|V(G)| = n\}$, due to their simplicity, and for each property set investigated, we will analyze both the size of the possible world and the extent of similarity for the graphs in the same possible world. But remember that in general, the set $\mathcal{P}$ in Definition 6 could contain any properties of G.

Let's analyze $PW(G, \mathcal{V}, \mathcal{P}_1)$ first. Suppose $|V_i| = k_i|B_i|$, where $k_i \geq 1$ is some positive integer. Then $G_{B,\mathcal{B}} \in PW(G, \mathcal{V}, \mathcal{P}_1)$ should satisfy the following two equations:

$$\sum_{i=1}^{m} k_i|B_i| = n \tag{1}$$

$$\sum_{i=1}^{m}\sum_{j=1}^{m} k_i k_j |B_i| CAdjM(\mathcal{B})[i:j] = 2m \tag{2}$$

Every feasible solution $(k_1, k_2, ..., k_m)$ with respect to Equation 1 and 2 corresponds to a valid $G_{B,\mathcal{B}}$. As a result, the number of feasible solutions with respect to Equation 1 and 2 then is the size of the possible world $PW(G, \mathcal{V}, \mathcal{P}_1)$ (note that G automatically satisfies Equation 1 and 2).

The analysis of $PW(G, \mathcal{V}, \mathcal{P}_2)$ is similar. The size of the possible world now becomes the number of feasible solutions $(k_1, k_2, ..., k_m)$ with respect to Equation 1.

Clearly, different possible world has different size, and thus sampling from different possible world may obtain graphs with quite different statistical properties, which may or may not be close to the desired properties of the original network. Therefore, the actual things the network publisher should do is then to find a proper tradeoff between anonymity and utility, by preserving the backbone and using a proper set $\mathcal{P}$ of property constraints which achieves moderately large $PW(G, \mathcal{V}, \mathcal{P})$ and acceptable utility. In the experimental part, we will see that even just using the property set P_2 provides a reasonable tradeoff.

We next propose the following graph backbone detection algorithm, as shown in Algorithm 2. Suppose G is a given graph and $\mathcal{V}$ is a sub-automorphism partition of G given. Here, G_V denotes the subgraph G induced by a subset V of $V(G)$, and $\mathcal{C}(G)$ denotes the set containing all the connected components of G. Let $V \in \mathcal{V}$, and let $v \in V$. The list $L(v)$ containing all the vertices u such that $u \in V$ and $N(v) \cap \bar{V} = N(u) \cap \bar{V}$, i.e., u and v shares the same neighbors outside V. For each $u \in L(v)$, we maintain a vertex pair (v, u) and we use $\mathcal{L}(V)$ to denote all such pairs between vertices in V. What's more, we use $G \cong_{\mathcal{L}} G'$ by restricting the isomorphism mapping between G and G' to map vertices conforming to the vertex pairs contained in $\mathcal{L}$. That is, if $v \in V(G)$ and $v' \in V(G')$, but $(v, v') \notin \mathcal{L}$, then v' is not considered to match v, even if actually there is such an isomorphism f mapping that $f(v) = v'$. Furthermore, by saying removing a subset V of $V(G)$ from G, we mean removing every vertex $v \in V$ from $V(G)$, and also removing all the edges induced by v from $E(G)$.

In order to test the isomorphism between the connected components in each cell of $\mathcal{V}$, with respect to $L(V)$, we simply check all the possible vertex mappings. For each mapping, we test the edge preservation that is required by an isomorphism. By this, we could then remove the duplicate connected components in each cell. Note that the removed components in each cell actually could be recovered through corresponding orbit copying operations, and thus the resulted graph is a reduction of G, with respect to $\mathcal{V}$. The resulted graph must be the backbone of the original graph simply because it could not be reduced any further.

We then propose a graph backbone based sampling strategy where the backbone is computed using the above algorithm.

Algorithm 2: Graph Backbone Detection

```
Input: $G$, $\mathcal{V}$
Output: $B_{G,\mathcal{V}}$
foreach $V \in \mathcal{V}$ do
    foreach $v \in V$ do
        Compute $L(v)$;
        foreach $u \in L(v)$ do
            Add $(v, u)$ into $\mathcal{L}(V)$;
        end
    end
    foreach $C \in \mathcal{C}(G_V)$ do
        if $\exists C' \in \mathcal{C}(G_V), G'_C \cong_{\mathcal{L}(V)} G_C$ then
            Remove $C'$ from $G$;
        end
    end
end
return The resulted graph, which must be $B_{G,\mathcal{V}}$;
```

Algorithm 3: Graph Backbone Based Sampling

```
Input: $G'$,$\mathcal{V}'$, $n = |V(G)|$, $SPD(1...|\mathcal{V}'|)$
Output: A connected subgraph $G_s$ of $G'$ such that $G_s \in PW(G, \mathcal{V}, P_2)$
Compute $B_{G',\mathcal{V}'}$;
$N = n - |V(B_{G',\mathcal{V}'})|$;
while $N > 0$ do
    Randomly pick $i$ with respect to $SPD(i)$ such that $(CPN(i) + 1) \cdot |B_i| \leq |V'_i|$, where $1 \leq i \leq |\mathcal{V}'|$,
    $B_i \in \mathcal{B}$ and $V'_i \in \mathcal{V}'$;
    $CPN(i) = CPN(i) + 1$;
    $N = N - |B_i|$
end
for $1 \leq i \leq |\mathcal{B}|$ do
    Repeat $Ocp(B_{G',\mathcal{V}'}, \mathcal{B}, B_i)$ $CPN(i)$ times;
end
return The resulted graph as $G_s$;
```

As shown in Algorithm 3, here $SPD(1...|\mathcal{V}'|)$ is an array containing the *sampling probability* from each cell of $\mathcal{V}'$. In general, $SPD(1...|\mathcal{V}'|)$ could be any probability distribution. However, since real social networks usually exhibit a heterogenous degree distribution, the number of vertices with larger degree will be less than the number of vertices with smaller degree. As a result, in $\mathcal{V}$, the size of cells that contain vertices with larger degree will be smaller, compared with those cells containing vertices with smaller degree. Hence, in practice we could tune SPD so that cells containing vertices with smaller degree will receive higher sampling rate (e.g.$SPD(i) = d_i^{-1} / \sum_{j=1}^{|\mathcal{V}'|} d_j^{-1}$, where d_i is the degree of those vertices contained in V'_i). We also maintain an array $CPN(1...|\mathcal{V}'|)$ to record the number of orbit copying operations that should be performed on each cell of the corresponding partition $\mathcal{B}$ of $B_{G',\mathcal{V}'}$. The basic idea of the algorithm is then to randomly copying the cells in $\mathcal{B}$ with respect to SPD, until the total number of vertices achieves $|V(G)|$. Note that the resulted graph may have a number of vertices slightly more than $|V(G)|$, but the additional number of vertices inserted will not exceed the size of the cell chosen at the last iteration of the while loop, which could usually be ignored since most cells in the automorphism partition of a real network are of very small size, compared with the whole population.

A big defect of this sampling strategy is its efficiency. We should first detect the backbone in the anonymized graph, which may be substantially larger than the original network. When processing each cell in Algorithm 2, we are indeed solving a set of graph isomorphism testing problems, whose complexity is still not determined and remains a open question. Specifically, neither we have found a polynomial time algorithm, nor we can prove that the problem is NP-complete. Therefore, in worst case, it is unlikely that there exists an efficient algorithm outperforming the brute-force search. What's more, the complexity of a series of orbit copying operations is $O(|V(B_{G',\mathcal{V}'})|^2)$, which also suffers heavy cost when the network becoming larger.

To avoid these computational overhead, in the following we propose an optimal sampling strategy with linear time complexity in worst case. This strategy tries to conserve the backbone as much as possible, in which we also use P_2 as the property set. As shown in Algorithm 4, here $S(1...\mathcal{V}')$ is an array that records the number of vertices should be sampled from each cell in $\mathcal{V}'$, with respect to SPD, $Visited(1...|V(G')|)$ and $Selected(1...|V(G')|)$ are two boolean arrays which will be used in the DFS procedure represented in Algorithm 5, and the other notations have the same meaning as they are in Algorithm 3.

Algorithm 4: Graph Backbone Approximated Sampling

```
   Input: $G'$, $\mathcal{V}'$, $n = |V(G)|$, $SPD(1...|\mathcal{V}'|)$
   Output: A connected subgraph $G_s$ of $G'$ such that $|V(G_s)| = n$
1  $N = n - |\mathcal{V}'|$;
2  while $N > 0$ do
3      Randomly pick $i$ with respect to $SPD(i)$ such that $S(i) < |V_i'|$, where $1 \leq i \leq |\mathcal{V}'|$ and $V_i' \in \mathcal{V}'$;
4      $S(i) = S(i) + 1$;
5      $N = N - 1$;
6  end
7  Uniformly pick a vertex $r \in V(G')$, and suppose $r$ in $V_j'$;
8  $Visited(r) = true$;
9  $Selected(r) = true$;
10 $S(j) = S(j) - 1$;
11 $n = n - 1$;
12 $DFS(r, n, Visited, Selected, S, \mathcal{V}')$;
13 return The subgraph of $G'$ induced by the vertices selected as $G_s$ (i.e., $G_s = G_V$ where
   $V = \{v | v \in V(G) \wedge Selected(v) = true\}$);
```

The main idea of the sampling procedure is to try to approximate the number of vertices in each cell of the original sub-automorphism partition $\mathcal{V}$. To make the sampling graph connected, we do a depth-first traversal on the graph G', as shown in Algorithm 5. Here v denotes the current vertex to be visited, and n is the remaining number of vertices that need to be sampled. We randomly select a vertex in G' as the root of the resulted DFS-tree. When testing whether a neighbor u of the current visiting vertex v should be selected, we check the remaining number of vertices that needs to be selected in the cell u belonging to. If the remaining number is greater than 0, we then select u and visit u recursively. This is illustrated from line 8 to 13 in Algorithm 5. We also keep track of the total remaining number n of vertices that need to be select, and before testing the selection of any vertex, we check whether $n < 1$ (from line 2 to 4 in Algorithm 5). If it is the case, then no more vertex needs to be chosen and the whole process terminates.

The philosophy here is that since G' preserves the backbone of G, the basic linkage pattern of G is reflected statistically more significant in G' than other linkage patterns. So when we run the random sampling procedure, the elements in the possible world has a higher chance to be obtained (or approximated) than other graphs outside the possible world. Actually, we may use more complicated randomizing strategy when sampling, to capture the backbone of G better. But as we shall see in the experimental part, the sampling strategy shown in Algorithm 4 already works quite well. Another advantage of the proposed sampling procedure is its efficiency. Since it is in fact a depth-first traversal of G' plus some preprocessing, the worst case running time is $O(V(G') + E(G'))$, which is linear.

Algorithm 5: $DFS(v, n, Visited, Selected, S, \mathcal{V}')$

```
Input: $v$, $n$, $Visited$, $Selected$, $S$, $\mathcal{V}'$
foreach $u \in N(v)$ do
    if $n < 1$ then
        return;
    end
    if $!Visited[u]$ then
        $Visited[u] = true$;
        //Suppose $u \in V_t'$.
        if $S(t) > 0$ then
            $Selected[u] = true$;
            $S(t) = S(t) - 1$;
            $n = n - 1$;
            $DFS(u, n, Visited, Selected, S, \mathcal{V}')$;
        end
    end
end
```

3.5 Experimental Results Of The Basic k-Symmetry Model

In this section, we give extensive studies on the k-symmetry model proposed. Three real network datasets, **Hep-Th**, **Enron** and **Net-trace** are used in our experiments, which are kindly provided by the authors of [7] and also used in the experimental evaluation of their work. Table 1 summarizes some statistics of the networks studied.

Table 1. Statistics of networks used.

Statistic	Network		
	Hep-Th	Enron	Net-trace
Number of vertices	2510	111	4213
Number of edges	4737	287	5507
Minimum degree	1	1	1
Maximum degree	36	20	1656
Median degree	2	5	1
Average degree	3.77	5.17	2.61

As in [7], here we consider an analyst who estimates a graph property by drawing sample graphs from G', measuring the property of each sample, and then aggregating measurements across samples. In other words, we only focus on the utility of the statistical properties of the original graph. We examine four properties commonly measured and reported on network data. *Degree* is a distribution of the degrees of all vertices in the graph. *Path length* is a distribution of the lengths of the shortest paths between 500 randomly sampled pairs of vertices in the network. *Transitivity* (or, *clustering coefficient* is a distribution of values where, for each vertex, we find the proportion of connected neighbor pairs among all possible neighbor pairs. *Network resilience* is measured by plotting the fraction of the number of vertices contained in the largest connected component as vertices are removed in decreasing order of degree [8].

We measured each of these characteristics for the original graph G and for a set of 20 output graphs produced by the sampling strategies with $k = 5$. We test both Algorithm 3 and 4. To some extent of our surprise, the results produced by the two strategies are almost the same, and what's more, the approximation algorithm (Algorithm 4) performs even a bit better than Algorithm 3 in the case of *Hep-Th* and *Net-trace* by introducing less new edges into the graph. This is due to the fact that Algorithm 3 will copy all of the links when deciding to sample a vertex, which may introduce many edges if a vertex with large degree (but unfortunately not included in the original graph) is decided to be copied. Therefore, due to the similarity of the results produced by the two algorithms, in Figure 8, we only show the results coming from the approximation strategy (Algorithm 4).

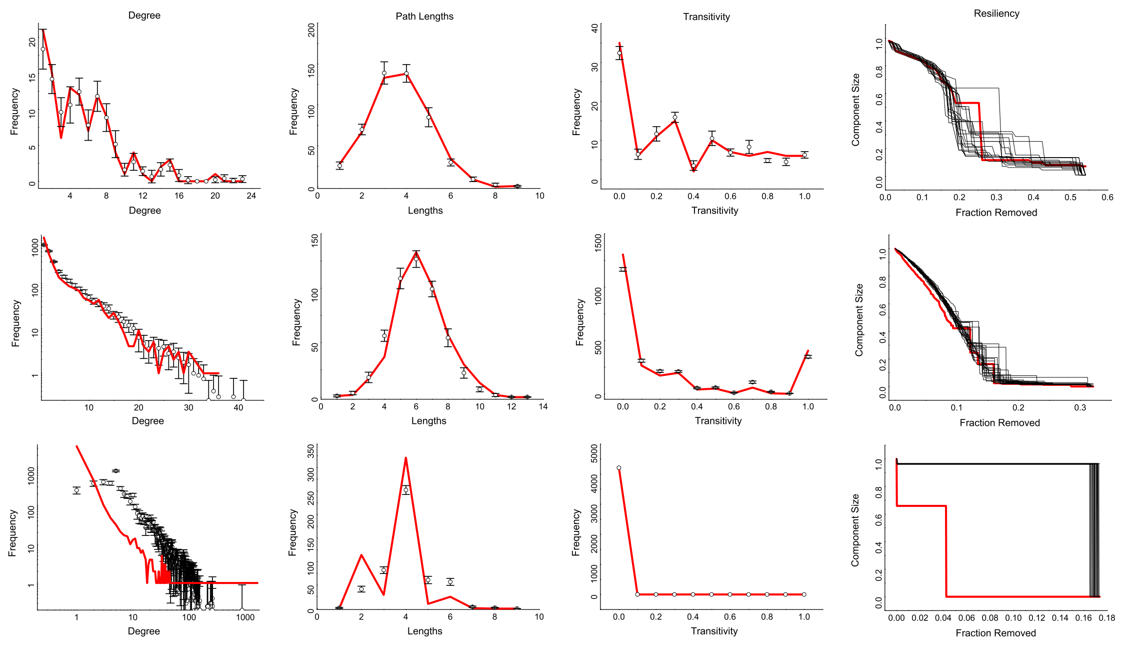


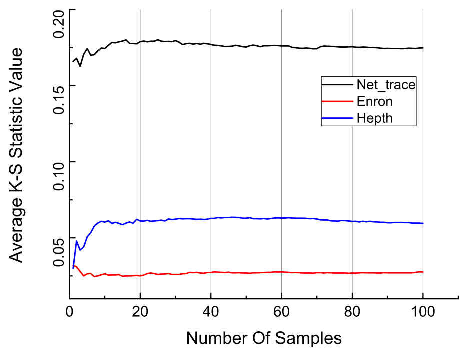
Fig. 8. Experimental results of the basic k-symmetry model

Discussion: As shown in Figure 8, in most cases except the case that the graph has an extremely heterogeneous degree distribution (e.g. the **Net-trace**, also see Table 1), the sampling strategies proposed perform quite well. The problem suffered from a extremely heterogeneous graph could be stated in an analytical way. If a graph has an extremely heterogeneous degree distribution, then it usually has some vertices with very large degree, and many vertices with very small degrees (say, 1 or 2), and what's more, vertices with large degrees are connected to many vertices with small degrees, to consume the degrees and form the graph (recall the well known result that $\sum_{v \in V(G)} d(v) = 2|E(G)|$). As mentioned above, these vertices with huge degree are very likely to be contained in a trivial orbit (namely, with cardinality 1), and those vertices with very small degrees are very likely to be contained in orbits with much larger cardinalities. To make the graph k-symmetric, according to our anonymization procedure, it is then very likely that we will enlarge the former orbits each by k times, but leave the later orbits untouched, especially when k is set to be small (say, 5). When we use the sampling procedure to get a sample graph, it is much harder for us to predict the size of the orbit containing vertices with small degrees, since their degrees are increased by the anonymization procedure but those vertices with large degrees are not.

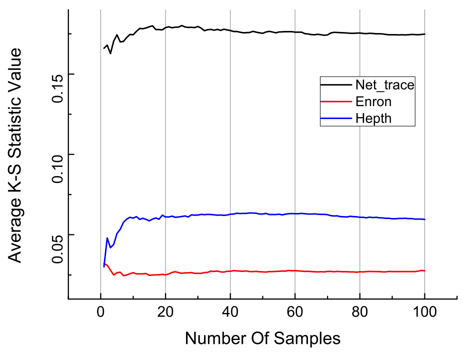
We also do the above experiments by setting $k = 10$, which gives similar results and thus we omit the details here to save space.

An interesting question is, when the number of sampling graphs increases, what is the behavior trend that these utility measures will follow? Would they keep on improving until converging to a steady point which may or may not match the actual properties of the original graph? Or they rather behaves in a random way so that the result would oscillate? It seems very difficult to give theoretical analysis at present and thus we conduct a further experimental study to probe this question.

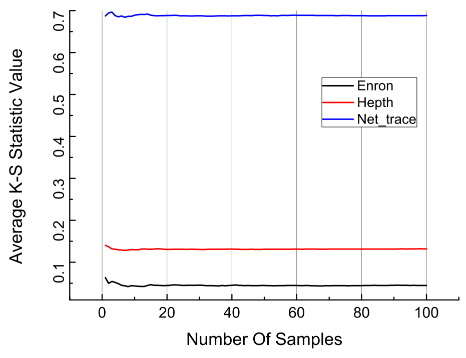
Here, we use the Kolmogorov-Smirnov statistic in the case of degree and shortest path distribution to test the utility change, which measures the maximum vertical distance between two cumulative distributions. The smaller this statistic is, the better the sampled graphs match the original graph on the compared distribution. We test the average value of this statistic on the two distributions considered, by increasing the number of sampling graphs from 1 to 100. Figure 9 gives the results.



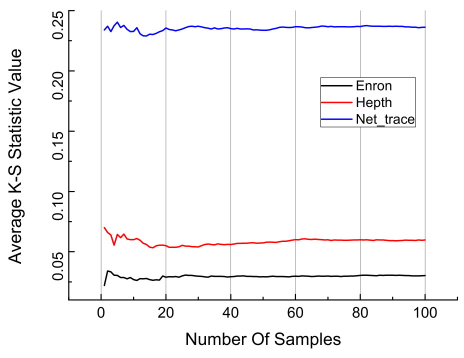
(a) Degree Distribution($k = 5$)



(b) Shortest Path Distribution($k = 5$)



(c) Degree Distribution($k = 10$)



(d) Shortest Path Distribution($k = 10$)

Fig. 9. Utility trend when the number of sampling graphs increasing

As shown in Figure 9, in all the cases tested, the value of the utility measure used will converge to a steady value quickly. It's a strong proof of the efficiency and reliability of our sampling method. We thus could achieve a reasonably good approximation to the original graph's properties by sampling a very small set of subgraphs from the anonymized network.

4 Variants of The Basic *k*-Symmetry Model

As we discussed in the last section, the proposed sampling procedure suffers from the extremely heterogeneous degree distribution of some graph. Actually, heterogeneous degree distribution is a property shared by most real world networks. The difference is just to what extent the heterogeneity would be. But in social networks, vertices with extremely large degrees are usually to be some well-known hubs. For example, in an email network of a company, the most connected vertex is very likely to be the email box address of the CEO. Here, we argue that these hubs are not deserved the effort to protect them from re-identification from revealed social network. There are three main reasons for this argument:

1 The hubs are well-known individuals. The relationships between hubs are very likely to be also well-known to the public (say, the communication between CEO and VP, in the previous example), even without the publication of the network. So there is no privacy leak(or even no privacy) in such cases.
2 The ultimate goal of privacy protection in social network is to block inferring relationships between individuals. Even the adversary knows the individuals represented by hubs, he cannot inspect relationships easily between hubs and un-hubs if any un-hub has at least k counterparts.
3 Technically, to make each hub have at least k counterparts, much more edges should be added, which make the published network much denser than the original graph.

Based on the above argument, in this section we will modify the basic k-symmetry model to allow no-copy for hubs in the graph. In the experimental part, we will see that such variation will improve the utility of the sampled graph.

4.1 *k*-Symmetry Model Excluding Hubs

In this version of the k-symmetry model excluding hubs, the only modification to Algorithm 1 is to decide which orbits containing hubs. Actually here, hubs mean those heavily connected vertices in the network. We should give an explicit definition for the term *heavily connected*. Formally, let $G = (V(G), E(G))$, and suppose the degree distribution of G is $P_G(d)$. Then the expectation and variance of $P_G(d)$ is

$$Exp_G(d) = \sum_{1 \le d \le MD(G)} d \cdot P_G(d) = \frac{1}{|V(G)|} \sum_{1 \le d \le MD(G)} d \cdot Freq_G(d),$$

$$(Var_G(d))^2 = Exp((d - Exp_G(d))^2) = \frac{1}{|V(G)|} \sum_{1 \le d \le MD(G)} (d - Exp_G(d))^2,$$

where $MD(G)$ is the maximum degree value and $Freq_G(d)$ is the frequency of the degree value d.

Now let $\delta \ge 0$, we specify a threshold $t = Exp_G(d) + \delta \cdot Var_G(d)$ and treat those vertices with degree exceeding t as hubs. Note that the value of δ could help us to control the threshold t, which again controls the number of vertices that are excluded from protection consideration. This makes our model more flexible for the consideration of the trade-off between anonymization and utility.

Based on this mechanism, our k-symmetry model excluding hubs is now simply the basic model plus a test of orbits containing hubs. Specifically, for any orbit C of the automorphism partition of the original graph G, we first see whether C is trivial. if it is, we than compare the degree d_C of the vertices (recall that all the vertices in C have the same degree) in C and the threshold t. If $d_C > t$, then we eliminate C from further processing and consider the next orbit.

4.2 Experimental Results of Excluding Hubs

In this section, we give experimental results of the variant model which excludes protection on some hub vertices. We only consider the *Net-trace* network since its degree distribution is extremely heterogeneous and the basic k-symmetry model doesn't perform well. We first investigate the relationship between the anonymization cost (i.e., the number of new vertices and edges inserted) and the percentage of vertices not protected.

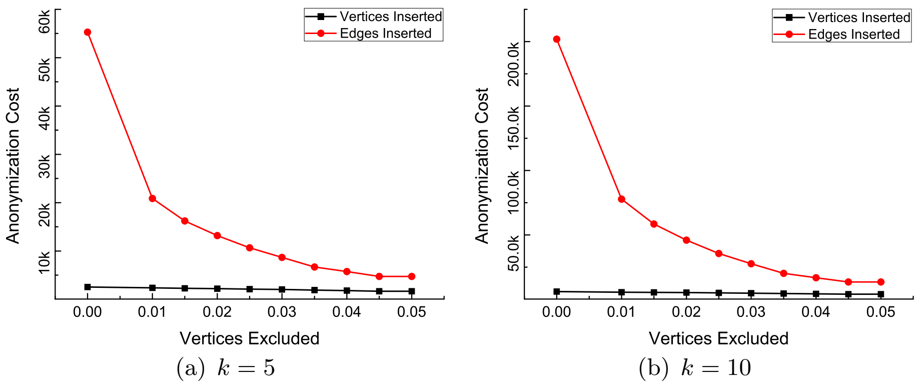


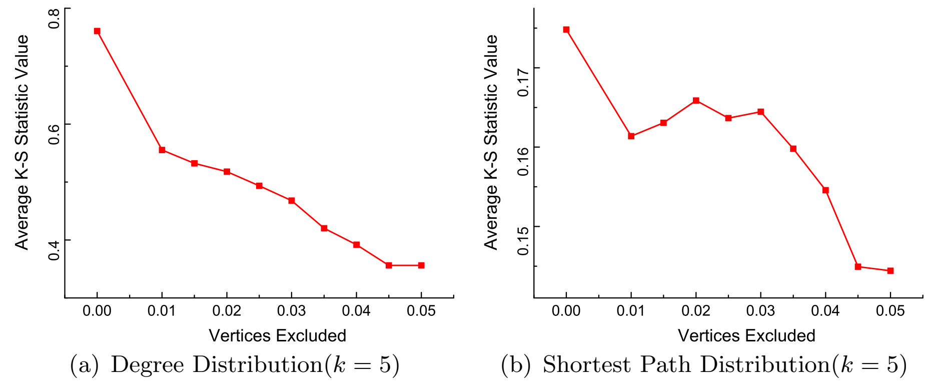
Fig. 10. Anonymization cost when some hub vertices are excluded from protection

Since the number of edges inserted dominates the overall overhead, we then ignore the cost of the new vertices introduced in the following discussion. As shown in Figure 10, when the fraction of vertices excluded (in degree decreasing order) increases slowly, the anonymization cost decreases dramatically. For instance, when $k = 10$, if 5% of vertices with largest degrees are excluded from protection, then the number of inserted edges decreases from 201,913 to 13,444, saved nearly 94% overhead. What's more, 61.5% overhead will be saved (the number of edge inserted decrease from 201,913 to 77,749) even if we just not consider protecting only 1% hub vertices, which is an impressive achievement.

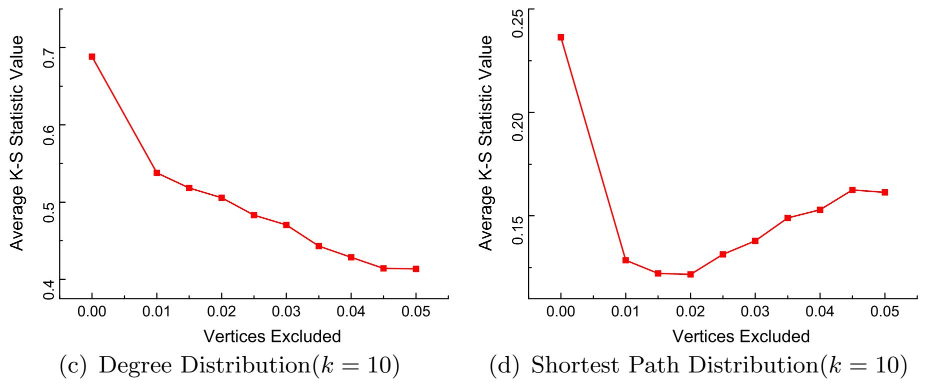
Another question is about the utility. Intuitively, since we now introduce less vertices and edges into the graph, the sampled graphs will now approximate the original graph more accurately. Figure 11 demonstrates this intuition.

Here we also use the average Kolmogorov-Smirnov statistic value as the measure of the utility quality. We still test this statistic on the degree and shortest path distribution. Since we have previously shown that this statistic will converge to a steady value fast when the number of samples increases, through a detailed experimental evaluation, we thus now simply use the value when the number of sampling graphs is 100 as the representative.

Don't forget that, the anonymity power of the model is not degraded much since now all vertices except some hub vertices still has at least $k - 1$ structural equivalent counterparts in the network, and the number of vertices excluded are very small, compared to the overall population.



(a) Degree Distribution($k = 5$) (b) Shortest Path Distribution($k = 5$)



(c) Degree Distribution($k = 10$) (d) Shortest Path Distribution($k = 10$)

Fig. 11. Utility improvements when excluding some hub vertices

4.3 A More Generalized Model

In this section, we propose a more generalized model, which simply specifies different size-constraints for different orbits. Formally, let $Orb(G) = \{V_1, V_2, ..., V_m\}$ be the automorphism partition of graph G, and d_i be the degree of vertices in the orbit V_i. We define an orbit size constraint function $f : Orb(G) \to I^+$, with two constraints. One is that $1 \leq f(V_i) \leq k$, where $k \geq 1$ is a specified positive integer, and the other is that f should be an non-increasing function with respect to d_i, that is, if $d_i \geq d_j$ then $f(V_i) \leq f(V_j)$. This generalized model provides further flexibility for the network publisher. The publisher could then test different fs and choose the one with which the sampling procedure on the anonymized graph could achieve the best utility results.

5 Discussion And Remarks

In this section, we discuss two problems related to all the models that we've proposed. The first problem is the computation of the automorphism partition of the original graph G, and the second problem is the optimization of the model. We shall give some remarks to each of these two problems.

5.1 Computation of The Automorphism Partition

It has been shown that the problem of determining the automorphism partition of a graph is polynomially equivalent to the graph isomorphism problem GI [9]. It's well known that the complexity of GI has not been determined. Specifically, $GI \in NP$, but neither we know it is in P, nor we can prove that it is *NP-complete*. But in practice, very efficient algorithms exist that could solve GI quickly even for relatively large graphs. The famous program `nauty` is one of these [10], whose implementation is based on the algorithm discussed in [11]. What's more, `nauty` also provides the functionality to compute the automorphism group of the given graph so the automorphism partition could also be determined. For extremely large graphs, `nauty` may not scale well. In such cases, a general approach called *graph stabilization* [12] may be used to obtain a good approximation to the automorphism partition of the graph. Specifically, the vertex partition named *total degree partition* $\mathcal{TDV}(G)$ which is the unique coarsest equitable partition finer than the *degree partition* (i.e., the partition obtained by grouping vertices according to their degrees), provides a good approximation to $Orb(G)$. Since the total degree partition could be determined in time $O(n^3 \log(n))$ by using the graph stabilization method(in practice it will usually be much faster than this), where n is the number of vertices in the graph, and since some approximation is acceptable for our purpose, then we could use $\mathcal{TDV}(G)$ to replace $Orb(G)$ when computing $Orb(G)$ seems very inefficient. In fact, although in general, $\mathcal{TDV}(G) \neq Orb(G)$, but to our surprise, in the real networks we've studied, no counterexample has been found. In other words, in all the real network we've studied, the total degree partition is just the same as the automorphism partition!

5.2 Minimizing The Number Of New Vertices Introduced

A quick look at Figure 1 gives us the following impression that it seems we introduce more vertices than required to achieve the k-symmetry constraint. For example, in Figure 1, when $k = 3$, the orbit $V_1 = \{v_1, v_2\}$ is copied once and results a new cell $V_1' = \{v_1, v_2, v_1', v_2'\}$. But if we just introduce the vertex v_1', the resulted cell $V_1'' = \{v_1, v_2, v_1'\}$ also satisfies the k-symmetry constraint. In other words, the vertex v_2' is not necessary to be introduced. Then the question how to achieve the k-symmetry constraint by introducing the *minimal* number of vertices arises. The ultimate observation here is that in Figure 1, v_1 and v_2 could be copied to each other. Based on the concept of graph backbone, we could answer the above question that if we first compute the backbone $B_{G,Orb(G)}$ of the original graph G, with respect to $Orb(G)$, and then apply the anonymization procedure (Algorithm 1) on $B_{G,Orb(G)}$, the introduced new vertices are then minimized, since in any cell of the associated sub-automorphism partition of $B_{G,Orb(G)}$, no two vertices could be copied to each other.

6 Related Work

The problem of protection of privacy in social networks was first proposed in [6], where they demonstrates that the naive anonymization strategy is not sufficient by studying both the active and passive model in depth. While active attacks are actually hard to carry out in many real social networks, passive attacks are much easier to do and thus have been more extensively studied. Some researchers focus on measures of anonymity (e.g. [13] and [14]), and others concerns various anonymization techniques. In [15], a technique based on random edge deletions and insertions was proposed, which is effective to resist some kind of attacks but suffers a significant cost in utility. Edge randomization techniques are further explored in [16], whose goal is to preserve the spectral properties. While the network utility is much improved, the effect on anonymity is not quantified. Other anonymization techniques based on the classic framework of k-*anonymity*([17], [18] and [19]) which is widely adopted in the privacy preserving release of traditional tabular data, have also been proposed. Zhou et al. [20] introduce a method to insert edges into the network until any vertex has a local neighborhood which is isomorphic to at last $k-1$ others. Liu et al. [21] present an efficient algorithm to make the network k-degree anonymous (i.e., for each vertex, there are at least $k-1$ other vertices sharing the same degree), also by inserting edges into the network. A variant version which allows simultaneously inserting and deleting edges from the network are also mentioned in [21], to achieve better utility. Both of these methods restrict the background structural knowledge to be the local properties of the target, and the utility problem is not handled very well since new introduced edges may significantly change the topology of the original network. To address these problems, Hay et al. [7] generalize the notion of background structural knowledge based attack model and propose an anonymization technique which first partitions the vertex set into subsets with size at least k and then publishes a generalized network on the partition level. They also proposed a strategy to achieve good approximation of the statistical properties of the original network, by sampling a set of graphs with same number of vertices and edges as the original network from the generalized graph, measuring their property values and then taking the average. But the time required by the anonymization algorithm is somewhat long. As reported in [7], on the network **Net-trace** with only 4213 vertices and 5507 edges, a few hours are needed for the algorithm to produce the result. This may be due to the large number of iteration steps used by the algorithm to search for an optimal vertex partition.

7 Conclusion And Future Work

This paper proposes a new framework to address the problem of protecting privacy of individuals in the published social network, where arbitrary background structural knowledge of individuals is considered possible to be used by the adversary. The proposed anonymization algorithm is efficient to produce a k-symmetric version of the original network in which the probability to re-identify any individual is at most $\frac{1}{k}$. The utility problem is also carefully studied, and two effective graph backbone based sampling strategies have been proposed. Experimental evaluation demonstrates both the efficiency of the anonymization procedure, and the good approximation of the original network achieved by the sampled graphs.

As future work, it is firstly interesting to see that how to extend the current model to achieve higher granularity of protection against more powerful attack model such as active attack. In privacy protection techniques of traditional tabular data, more powerful framework such as *l-diversity*[22] has been proposed and it is worth the effort to investigate whether the current model could absorb some ideas from these techniques. Secondly, since k-symmetry model is a general framework, it is also very interesting to seek other algorithms which could make a network k-symmetric, with different restrictions on the operations allowed (e.g., edge insertion only or simultaneously edge insertion and deletion). Thirdly, given a network, how to choose a proper set of property constraints to achieve the best tradeoff between anonymity and utility is an important open problem. Fourthly, the sampling procedures as shown in Algorithm 3 and 4 are just two of the possible random sampling strategies and it is an interesting problem to try other strategies as well. Finally, the notion of graph backbone proposed in this paper may be used in the solution of other graph related applications since it reflects the basic linkage pattern of the original graph.

References

1. D. Watts and S. Strogatz, “Collective dynamics of ’small-world’ networks,” *Nature*, vol. 393, pp. 440–442, 1998.
2. A. Barabasi and R. Albert, “Emergence of scaling in random networks,” *Science*, vol. 286, 1999.
3. M. E. J. Newman, “Assortative mixing in networks,” *Phys. Rev. Lett*, vol. 89, no. 208701, 2002.
4. C. Song, S. Havlin, and H. A. Makse, “Self-similarity of complex networks,” *Nature*, vol. 433, p. 392, 2005.
5. J. J. Potterat, L. Phillips-Plummer, S. Q. Muth, R. B. Rothenberg, D. E. Woodhouse, T. S. Maldonado-Long, H. P. Zimmerman, and J. B. Muth, “Risk network structure in the early epidemic phase of hiv transmission in colorado springs,” *Sexually Trans. Infections*, 2002.
6. L. Backtrom, C. Dwork, and J. Kleinberg, “Wherefore art thou r3579x? anonymized social networks, hidden patterns, and structural steganography,” in *WWW’07*, 2007.
7. M. Hay, G. Miklau, D. Jensen, D.Towsley, and P. Weis, “Resisting structural re-identification in anonymized social networks,” in *VLDB’08*, 2008.
8. R. Albert, H. Jeong, and A.-L. Barabasi, “Error and attack tolerance of complex networks,” *Nature*, vol. 406, p. 378, 2000.
9. R. Mathon, “A note on the graph isomorphism counting problem,” *Information Processing Letters*, vol. 8, pp. 131–132, 1979.
10. P. Foggia, C. Sansone, and M. Vento, “A performance comparison of five algorithms for graph isomorphism,” in *the 3rd IAPR TC-15 Workshop on Graph-based Representations in Pattern Recognition*, 2001.
11. B. McKay, “Practical graph isomorphism,” *Congressus Numerantium*, vol. 30, pp. 45–87, 1981.
12. M. Klin and G.Tinhofer, “Algebraic combinatorics in mathematical chemistry. methods and algorithms. iii. graph invariants and stabilization methods (preliminary version),” Technische Universitat Munchen, Tech. Rep. TUM-M9902, 1999.
13. L. Singh and J. Zhan, “Measuring topological anonymity in social networks,” in *Intl. Conf. on Granular Computing*, 2007.
14. D. W. Wang, C. J. Liau, and T. S. Hsu, “Privacy protection in social network data disclosure based on granular computing,” in *International Conference on Fuzzy System*, 2006.
15. M. Hay, G. Miklau, D. Jensen, P. Weis, and S. Srivastava, “Anonymizing social networks,” UMass Amherst, Tech. Rep. 07-19, 2007.
16. X. Ying and X. Wu, “Randomizing social networks: a spectrum preserving approach,” in *SIAM Conf. on Data Mining*, 2007.
17. P. Samarati and L. Sweeney, “Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression,” SRI International, Tech. Rep., 1998.
18. P. Samarati, “Protecting respondent’s privacy in microdata release,” *IEEE Transactions on Knowledge and Data Engineering*, 2001.
19. L. Sweeney, “k-anonymity: a model for protecting privacy,” *Journ. of Uncertainty, Fuzziness, and KB Systems*, 2002.
20. B. Zhou and J. Pei, “Preserving privacy in social networks against neighborhood attacks,” in *ICDE’08*, 2008.
21. K. Liu and E. Terzi, “Towards identity anonymization on graphs,” in *SIGMOD’08*, 2008.
22. A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, “l-diversity: Privacy beyond k-anonymity,” in *ICDE’06*, 2006.