The Anonymous Subgraph Problem

Andrea Bettinelli

DTI, Universit degli Studi di Milano, via Bramante 65, 26013 Crema (CR), Italy

Leo Liberti

LIX, cole Polytechnique, F-91128 Palaiseau, France

Franco Raimondi

School of Engineering and Information Sciences, Middlesex University, London, UK

David Savourey

LIX, cole Polytechnique, F-91128 Palaiseau, France

Abstract

In this work we address the Anonymous Subgraph Problem (ASP). The problem asks to decide whether a directed graph contains anonymous subgraphs of a given family. This problem has a number of practical applications and here we describe three of them (Secret Santa Problem, anonymous routing, robust paths) that can be formulated as ASPs. Our main contributions are (i) a formalization of the anonymity property for a generic family of subgraphs, (ii) an algorithm to solve the ASP in time polynomial in the size of the graph under a set of conditions, and (iii) a thorough evaluation of our algorithms using various tests based both on randomly generated graphs and on real-world instances.

Keywords: anonymity, anonymous routing, Secret Santa, graph topology

1. Introduction

Given a directed graph $G = (V, A)$, many problems can be modelled as the search for a subgraph $S \subseteq A$ with specific properties. There are

Email addresses: andrea.bettinelli@unimi.it (Andrea Bettinelli), liberti@lix.polytechnique.fr (Leo Liberti), f.raimondi@mdx.ac.uk (Franco Raimondi), savourey@lix.polytechnique.fr (David Savourey)

applications in which it is desirable to ensure S is anonymous, in the sense described below. In this work we formalize an anonymity property for a generic family of subgraphs and the corresponding decision problem. We devise an algorithm to solve a particular case of the problem and we show that, under certain conditions, its computational complexity is polynomial. We also examine in detail several specific family of subgraphs.

This problem has not been previously studied in its general form, but other anonymity-related problems have received great attention in the last years. For instance, the rising popularity of on-line communities and social networks has motivated the analysis of their structures. While this can explain interesting aspects of human social behavior, it also creates privacy concerns. A social network is usually modelled as a graph, where vertices represent individuals and edges correspond to relationships between individuals. If such graph is released to the public, we must guarantee the privacy of the users is preserved. The simplest graph-anonymization technique consists in removing the identities of the vertices, replacing them with random identification numbers. Recently Backstrom et al. [1] have shown that this does not always guarantee privacy and that there exist adversaries able to identify target individuals and the relationships between them by solving a set of restricted graph isomorphism problems. Hay et al. [2] propose a definition of graph anonymity: a graph satisfies *k-candidate anonymity* if every vertex shares the same neighborhood structure with at least $k-1$ other vertices. However, the authors concentrate on the formulation of anonymity definitions and not on the design of algorithms that guarantee to obtain a graph which satisfies them. Zhou and Pei [3] consider the following definition of graph-anonymity: a graph is k-anonymous if for every vertex there exist at least $k-1$ other vertices that share isomorphic 1-neighborhoods. They also consider the problem of minimum graph-modification required to obtain a graph that satisfies the anonymity requirement. Zheleva and Geetor [4] study the problem of protecting sensitive links between individuals in an anonymized social network and propose simple edge-deletion and vertex-merging algorithms to reduce the risk of sensitive link disclosure. In [5] Frikken and Golle study the problem of privately assembling a graph whose pieces are owned by different parties. They propose a set of protocols that allow to reconstruct the graph without revealing the identity of the vertices. Feder et al. [6] introduce another definition of graph anonymity: a graph is (k,l)-anonymous if for every vertex in the graph there exist at least k other vertices that share at least l of its neighbors. The authors propose an algorithm to compute the minimum number of edges to be added so that a given graph becomes (k,l)-anonymous.

Another field of research is the development of techniques to measure the level of anonymity provided by a system. Chaum [7] introduces the notion of anonymity set as the set of participants who are likely to be senders (or recipients) of a message. The larger is the set, the stronger is the anonymity of its members. Serjantov and Danezis [8] show that the size of the anonymity set is inadequate for expressing instances where not all members are equally likely to have sent a particular message and they propose an *effective anonymity set size*, based on the information theoretic concept of entropy. They interpret it as the amount of additional information the attacker needs to identify a user. A similar entropy-based metric has been independently proposed by Diaz et al. [9]. Tóth and Hornàk [10] introduce the notion of *source-hiding* and *destination-hiding*. A system is source-hiding with parameter Θ if the attacker cannot assign a sender to any message with probability greater than Θ. In a similar way, a system is destination-hiding for a given parameter Ω if the attacker cannot determine the recipient of any message with probability greater than Ω. The authors also show that a system may appear near optimal with respect to the entropy-based metrics even if the attacker can identify the sender or the recipient of some messages with high probability. Newman et al. [11] propose to use an entropy-based approach to evaluate the level of protection provided by a Traffic Analysis Prevention system. Instead of computing the size of the anonymity set for each message, Edman et al. [12] use a combinatorial approach to obtain a metric that considers all messages simultaneously. Gierlichs et al. [13] generalize the ideas of Edman et al. to take into account multiple messages sent or received by the same user and propose an algorithm to compute this metric.

In Meurdesoif et al. [14] a situation almost opposite to ours is tackled: to find a minimum cost set of arcs that allows to completely identify any path between fixed source and destination vertices.

This paper is organized as follows. In Section 2 we introduce the definition of an anonymous family of subgraphs and we formalize the Anonymous Subgraph Problem. In Section 3 an algorithm to solve a restriction of the problem is described. Two applications of this restriction of the problem are illustrated in Sections 4 and 5; for a particular case of the latter, a more efficient algorithm is also derived. In Section 6 we return to the general ASP problem and we present an application of the general problem and an ad-hoc algorithm is proposed for solving it. Finally we draw some conclusions in Section 7.

2. Characterization of anonymity

In this section we start by providing a formalization of the concept of anonymity for a family of subgraphs of a graph G (Defn 1) and by introducing the corresponding decision problem (Defn 2). We then describe a restriction of this problem (Defn 3) that can be solved in polynomial time.

Given a digraph $G = (V, A)$, let $|V| = n$ and $|A| = m$. We say that $G' = (V', A')$ is a subgraph of G when $V' \subseteq V$ and $A' \subseteq A$. Since a subset of arcs of A uniquely induces a subgraph of G modulo isolated vertices, we sometimes employ the word "subgraph" to mean a set of arcs inducing a subgraph. We characterize anonymity of a subgraph S, meant as a subset of arcs in G, so that no partial information about the topology of S may be used to infer the information that another arc of A is in S. We do this by requiring G to have a family $\mathcal{Y}$ of subgraphs, that includes S itself, such that for every choice of partial information $C \subset S$ and any arc $a \in S \setminus C$, there is always another member $S' \neq S$ of the family that includes C but not a. Thus, C never uniquely determines S within the family.

We call C a *partial view* of S. Let $\mathscr{P}(A)$ denote the set of all subsets of A, and let $P_V : \mathscr{P}(A) \times \mathcal{Y} \to \{0, 1\}$ be the function

$$P_V(X, S) = \begin{cases} 1 & X \text{ is a partial view of } S \\ 0 & \text{otherwise} \end{cases}$$

that defines which subsets are considered a partial view of a certain subgraph.

Intuitively, a family of subgraphs $\mathcal{Y}$ is anonymous if the knowledge of a partial view C of any subgraph S of $\mathcal{Y}$ does not lead to infer the presence of another arc b in S. We formalize this by means of the following:

Definition 1 (Anonymous family of subgraphs). Given a digraph $G = (V, A)$, a family of subgraphs $\mathcal{Y} \subseteq \mathscr{P}(A)$ and a function $P_V : \mathscr{P}(A) \times \mathcal{Y} \to \{0, 1\}$, $\mathcal{Y}$ is *anonymous* in G if

$$\forall S \in \mathcal{Y}, \forall C \in \{X | P_V(X, S) = 1\}, \ \forall b \in S \setminus C \quad \exists T \in \mathcal{Y} : \ C \subseteq T \wedge b \notin T.$$

We call *anonymous subgraphs* the elements of an anonymous family $\mathcal{Y}$. Intuitively, Definition 1 captures the standard notion of ignorance described, for instance, in [15]: an agent ignores a fact if, given its current information, the agent cannot establish whether this fact is true or false. In Definition 1, C is the information an agent has, and the definition of anonymity requires that, given this information, the agent cannot infer anything about any arc b not in C.

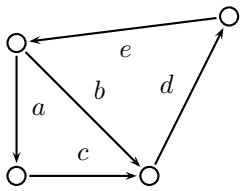

Figure 1: Example of a graph and subgraph families.

Example 1. Consider the directed graph in Fig. 1 and the following partial view function

$$P_V(X,S) = \begin{cases} 1 & X \subseteq S \wedge |X| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

that is, at most one arc of the subgraph is known. For a subgraph $S = \{a, b\}$ there are three partial views to be considered: $\emptyset$, $\{a\}$ and $\{b\}$. The family of subgraphs $\mathcal{X} = \{\{a, b\},\ \{b, c\},\ \{a, d\}\}$ is clearly not anonymous: the partial view $\{c\}$ allows to determine that the subgraph contains also arc b. On the contrary, the family $\mathcal{Y} = \{\{a, b\},\ \{b, d\},\ \{a, d\}\}$ satisfies the anonymity conditions.

It is now possible to define the *Anonymous Subgraph Problem* (ASP) as the decision problem of checking if a family of subgraphs contains an anonymous family with respect to a given graph.

Definition 2 (Anonymous Subgraph Problem). Given a digraph $G = (V, A)$, a family $\mathcal{F}$ of subgraphs of G and a function $P_V : \mathscr{P}(A) \times \mathcal{F} \to \{0, 1\}$, is there a non empty subset $\mathcal{Y}$ of $\mathcal{F}$ which is anonymous in G?

We now consider a special case of Defn. 1 when the set of partial views of each subgraph S is defined by $\{C |\, C \subseteq S \wedge |C| = 1\}$, i.e. only one arc of the subgraph is known, obtaining the following definition of anonymity:

Definition 3. Given a digraph $G = (V, A)$, a family of subgraphs $\mathcal{Y} \subseteq \mathscr{P}(A)$ is *anonymous* in G if the knowledge of any edge a from any subgraph S of $\mathcal{Y}$ can not lead to infer another edge b of S. This is the following formula:

$$\forall S \in \mathcal{Y}, \forall a \neq b \in S \quad \exists T \in \mathcal{Y} :\ a \in T \wedge b \notin T.$$

In other words, if we only know that $a \in S$, we cannot infer that also $b \in S$, since there is a $T \in \mathcal{Y}$ such that $a \in T$ but $b \notin T$. We will refer to ASP1

to denote the Anonymous Subgraph Problem where Definition 3 is used to characterize anonymity. As we show in the following sections, this restricted version of ASP has a number of practical applications. In the next section we propose an algorithm to solve ASP1 and we show that under certain conditions its computational complexity is polynomial in the size of the graph G.

3. A polynomial time algorithm for ASP1

Algorithm 1 is a recursive algorithm that solves ASP1: it returns an element of $\mathcal{Y}$, if an anonymous $\mathcal{Y} \subseteq \mathcal{F}$ exists, and an empty set otherwise. The algorithm is based on the following observation: if there exist two distinct arcs $a, b \in A$ such that no subgraph $T \in \mathcal{F}$ contains a but not b, then all the subgraphs $S \in \mathcal{F}$ containing a form a non-anonymous family, since knowledge of a will lead to infer b for every arc set in the family. Thus, we can transfer the anonymity property from subgraphs to arcs. At the top level of recursion, arcs in P play the role of the arc a. The algorithm iteratively removes arcs from P. If no subgraphs can be found satisfying the additional constraints given by P, then the family is not anonymous in G.

We define the subroutine $\textsc{FindSG}(G, \mathcal{F}, P, X)$ as follows:

$$\textsc{FindSG}(G, \mathcal{F}, P, X) = \begin{cases} \text{lexicographically smallest member of} \\ \{S \in \mathcal{F} \mid X \subseteq S \subseteq P\} \text{ if latter is nonempty} \\ \\ \emptyset \quad \text{otherwise.} \end{cases}$$

Theorem 1. *Alg. 1 correctly solves the ASP1.*

Proof. First we observe that, if the algorithm returns a non-empty solution, at Line 7 the set $\mathcal{Y}$ of subgraphs in $\mathcal{F}$ where every arc belongs to P is anonymous in G. Let $S \subseteq P$ be an element of $\mathcal{Y}$, we know that $\forall a, b \in S \; \exists T \in \mathcal{Y}$ s.t. $a \in T$ and $b \notin T$, otherwise a would have been removed from P in Line 4. Assume now there is a solution to ASP1 and Alg. 1 returns $\emptyset$. The existence of a solution implies the existence of a non-empty anonymous set $\mathcal{Y}$. If Alg. 1 reached Line 7 with $\mathcal{Y} \subseteq \mathscr{P}(P)$, then by definition of $\textsc{FindSG}(G, \mathcal{F}, P, X)$, Alg. 1 would not have returned $\emptyset$. Thus we know that the algorithm reached Line 7 with $\mathcal{Y} \setminus \mathscr{P}(P) \neq \emptyset$. Consider the first time an arc $a \in A$ used in at least one element of $\mathcal{Y}$ was removed from P. At that time $\mathcal{Y} \subseteq \mathscr{P}(P)$, so a should not have been removed because $\forall b \neq a \in P \; \exists T \in \mathcal{Y}$ s.t. $a \in T$ and $b \notin T$. Moreover, since initially $|P| = m$ and at every recursive call the cardinality of P is decreased by one, we are sure that the number of recursive calls is bounded by m. □

At each call the subroutine FindSG (which solves the subproblem) is executed up to m^2 times. Thus, if FindSG has computational complexity $O(\gamma)$, the worst case complexity of the overall algorithm is $O(m^3\gamma)$. In conclusion, if we are provided a polynomial algorithm (in the size of $|G|$) to solve the subproblem, we can solve ASP1 in polynomial time.

Algorithm 1 Algorithm for solving the ASP1. The first call has $P = A$.

```
1: FindAnonymousSG(G, F, P):
2: for all a ≠ b ∈ P do
3:   if FindSG(G, F, P \ {b}, {a}) = ∅ then
4:     return FindAnonymousSG(G, F, P \ {a})
5:   end if
6: end for
7: return FindSG(G, F, P, ∅)
```

Definition 2 holds for a generic family $\mathcal{F}$ of subgraphs of G, which might have non-polynomial size in terms of $|G|$. In real applications we usually have to deal with a family $\mathcal{F}$ characterized by specific properties, such as, for example, sets of disjoint cycles. By exploiting these properties, we can describe $\mathcal{F}$ implicitly and, in some useful cases, obtain polynomial procedures to solve FindSG independently of $|\mathcal{F}|$.

4. Secret Santa Problem

In this section we present a first application of ASP1 to a practical problem that requires to reason about knowledge in a system formalized as a graph. More in detail, when in Defn. 3 the family $\mathcal{F}$ of subgraphs of a graph G is the set of all Vertex Disjoint Circuit Covers (VDCCs — see Defn. 4), we obtain the Secret Santa Problem described in [16] (notice that the definitions of anonymity used here are slightly different from those given in [16]).

The basic concept of the Secret Santa ritual is simple. All of the participants' names are placed into a hat. Each person picks randomly a recipient's name from the hat, keeps the name secret, and buys a gift for the named recipient. The label on the gift wrapping indicates the recipient's name but not the buyer's. All the gifts are then placed in a general area for opening at a designated time.

Additional constraints are considered in the definition of the problem: it may be required that self-gifts and gifts between certain couples of participants (e.g. siblings) should be avoided. The problem can be modelled with

a digraph, where vertices represent the participants and arcs (u, v) the fact that participant u is eligible to select v as a recipient's name. We want to determine if the topology of the graph allows an anonymous exchange of gifts, i.e. one having the property that nobody can discover any gift assignment knowing the graph topology and their own recipient names.

The problem can be formulated as an ASP1 where $\mathcal{F}$ is the family of all the VDCCs of the graph.

Definition 4. A *Vertex Disjoint Circuit Cover* (VDCC) for $G = (V, A)$ is a subset $S \subseteq A$ of arcs of G such that: (a) for each $v \in V$ there is a unique $u \in V$, called the predecessor of v and denoted by $\pi_S(v)$, such that $(u, v) \in S$; (b) for each $v \in V$ there is a unique $u \in V$, called the successor of v and denoted by $\sigma_S(v)$, such that $(v, u) \in S$. We denote by $\mathcal{C}$ the set of all VDCCs in G.

Since gifts must be exchanged anonymously, not all VDCCs are acceptable: e.g. when $V = \{1, 2\}$ and $A = \{(1, 2), (2, 1)\}$, there is a unique VDCC, so each person knows that the other person will make her a gift. Informally, we define a graph G as a Secret Santa graph if it admits at least one VDCC ensuring anonymity.

Definition 5. A graph G is a *Secret Santa graph* (SESAN) if there exists an anonymous family $\mathcal{Y}$ of VDCCs in G. Elements of $\mathcal{Y}$ are called acceptable solutions.

By the definition of anonymity (Def. 3), even if a participant knows his/her own gift assignment a, he/she does not gain any knowledge with respect to any other gift assignment b.

4.1. Examples

Consider the graph obtained by replacing each edge in $K4$ (the leftmost graph in Fig. 2) with two anti-parallel arcs. This graph is SESAN: the 6 subgraphs on the right in Fig. 2 are an anonymous family of VDCCs in $K4$.

Consider now the graph on the left in Fig. 3. By replacing each edge with two anti-parallel arcs we obtain another SESAN graph. It is enough to combine two independent solutions for the $K4$ components. However, not all VDCCs are acceptable solutions. The right hand side of Fig. 3 is a VDCC, but it does not guarantee anonymity: if the arc a is used, we are forced to include the arc b.

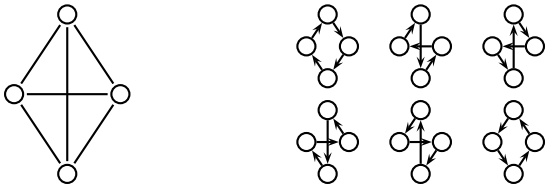

Figure 2: $K4$ and an anonymous family of VDCCs.

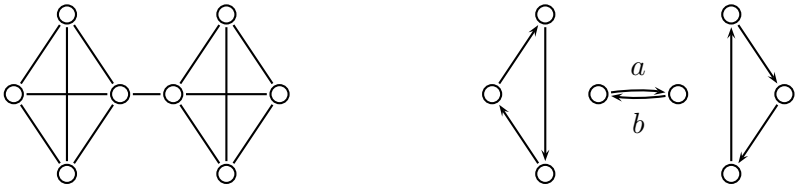

Figure 3: Example of SESAN graph and a non anonymous VDCC.

4.2. Solving the Secret Santa problem

As stated above, the secret Santa problem can be formulated as an ASP1 and, therefore, it is solved by Alg. 1. In this case FINDSG$(G, \mathcal{F}, P, \{(i,j)\})$ requires to find a VDCC with restrictions on the arcs that can be used. As shown in [16] it can be done in $O(n^{\frac{1}{2}}m)$ by solving an assignment problem on a bipartite graph $B = (U_1, U_2, A')$, where $U_1 = U_2 = V \setminus \{i,j\}$ and $A' = P$.

We generated groups of 20 random graphs with $|V| \in \{10i | 1 \leq i \leq 5\}$ and arc generation probability $p \in \{0.05i | 1 \leq i \leq 8\}$. The plot in Fig. 4 shows, for each $|V|$ and p the number of graphs out of 20 that are SESAN. We also used randomly generated power law graphs because they are a more realistic model of the structure of a social network [17]. In these graphs the expected degree of the i-th vertex is $\alpha n(i^{-t})$, where n is the number of vertices. Hence, an arc is created between the vertices i and j with probability $\frac{(\alpha n i^{-t})(\alpha n j^{-t})}{\sum_{k=1}^{n} \alpha n k^{-t}}$. We generated graphs with $\alpha = 0.5$, t in the range $[0.1, 0.4]$ and $|V| \in \{10i | 1 \leq i \leq 5\}$. The plot in Fig. 5 shows, for each $|V|$ and t, the number of graphs out of 20 that are SESAN. The results confirm the intuition that the SESAN property becomes more common as the size and the density of the graph increase.

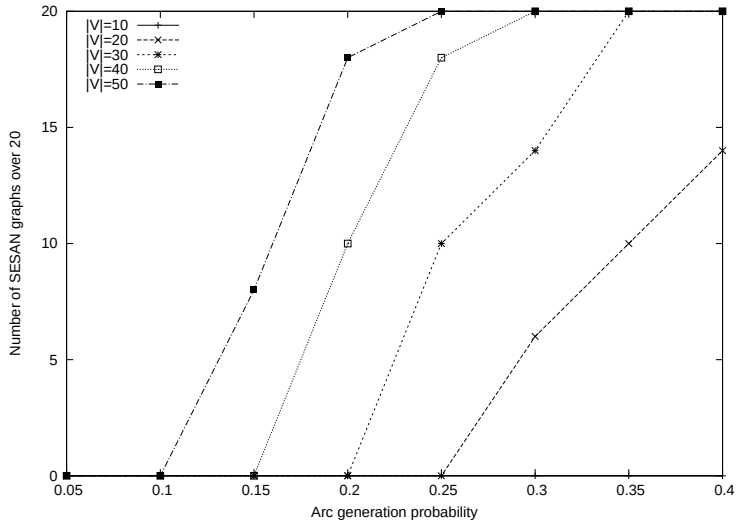

Figure 4: Proportion of SESAN random graphs with p ranging in $[0.05, 0.4]$.

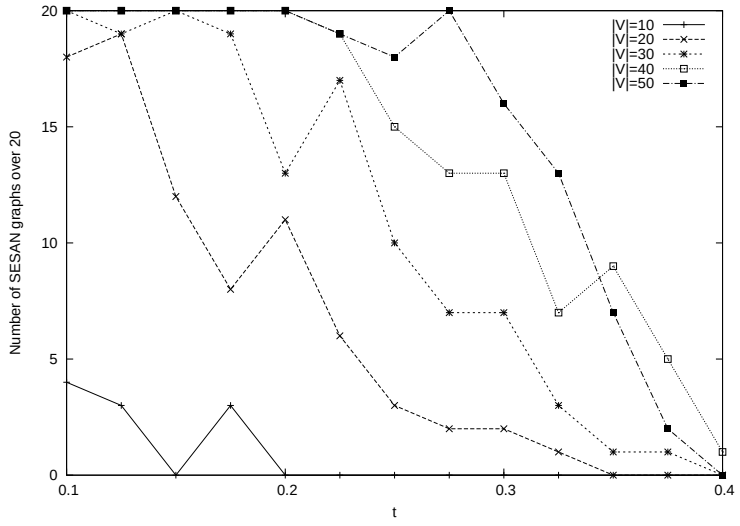

Figure 5: Proportion of SESAN random power law graphs with t ranging in $[0.1, 0.4]$ and $\alpha = 0.5$.

5. Anonymous routing

In this section we first provide two example of anonymous routing protocols. We then provide a formalization of anonymous routing in Section 5.2.

5.1. Examples of anonymous routing

In many contexts it is desirable to hide the identity of the users involved in a transaction on a public telecommunication network. Anonymous routing consists of guaranteeing anonymity and anti-localization of sender and/or receiver of messages in a communication network. It protects user communication from identification by third-party observers According to the specific application, we may be interested in:

- *sender anonymity* at any node (using local node information), at the receiver node (using receiver node information) or for a global attacker (who can monitor traffic on every link in the network);
- *receiver anonymity* at any node, at the sender node or for a global attacker;
- *sender-receiver unlinkability* (i.e. knowing that u sent a message and v received one is not sufficient to establish that u sent a message to v) at any node or for a global attacker.

In order to obtain anonymous routing, a protocol of communication must be defined among the users of the network. Several protocols have been proposed in the literature to provide anonymous routing features [18, 19, 20]. We now briefly describe two such protocols, Onion Routing and Crowds, to show how anonymous routing can be achieved. Then we show how ASP1 can be used to identify if the topology of a network is unsuitable to support an effective anonymous routing.

Onion routing is a general-purpose protocol [19] that allows anonymous connection over public networks. Messages are routed through a number of nodes called *Core Onion Routers* (CORs). In order to establish a connection, the initiator selects a random path through the CORs and creates an onion, a recursively layered data structure containing the necessary information for the route. Each layer is encrypted with the key of the corresponding COR. When a COR receives an onion, a layer is "unwrapped" by decrypting it with the COR's private key. This reveals the identity of the next router in the path and a new onion to forward to that router. Since inner layers are encrypted with different keys, each router obtains no information about the path, other than the identity of the following router. There are two

possible configurations for an end-user. They can either run their own COR (local-COR configuration) or use one of the existing ones (remote- COR). The first option requires more resources, but provides better anonymity.

Crowds is a system proposed by Reiter and Rubin [20] that aims to increase the privacy of web transactions by providing sender anonymity. The idea is to hide sender's actions within the actions of many others. To execute a web transaction a user first joins a "crowd" of other users. The request is first passed to a randomly selected member of the crowd. That member can either submit the message to the end server or forward it to another randomly chosen member of the crowd. Thus, when the request eventually reaches the end server, it is submitted by a random member of the crowd, preventing the end server from identifying its true initiator. Even the other members of the crowd cannot identify the sender, because it is not possible to distinguish a newly generated message from a forwarded one.

5.2. A formalization of anonymous routing

Attacks against anonymous routing protocols are usually based on traffic analysis. This means monitoring some of (or all) the links of the network and then try to correlate information in order to rebuild the path followed by messages. In some circumstances knowledge about a path can also be derived from the topology of the network. As an example, consider the graph in Fig. 6: there are several paths from vertex s to vertex t, but the presence of the arcs $(1, t)$ or $(2, t)$ in a path forces the presence of the arc $(s, 1)$ as well. We call this a "forced path", which we formalize as follows:

Definition 6 (Forced paths). A digraph $G = (V, A)$, contains a *forced path* if there exist two vertices $s, t \in V$ and two edges $a, b \in A$ s.t. if b belongs to a path from s to t, then also a belongs to the same path.

The presence of a forced path can be exploited by an attacker monitoring the traffic on those links to restrict the set of potential senders for the message. We introduce the notion of *strong path anonymity* to characterize topologies that do not have forced paths:

Definition 7 (Strong path anonymity). A digraph $G = (V, A)$ has the strong anonymity path property iff it does not contain forced paths (in the sense of Defn. 6).

Verification that a graph has the strong anonymity property can be reduced to solving an instance of ASP1 for each pair of nodes (s, t) of the network, where G is the graph representing the network and S is the family

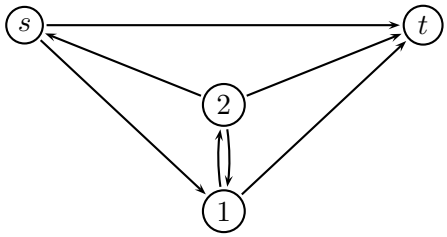

Figure 6: Example of non-anonymous family of paths.

of all paths of length at least 2 between the two nodes. We exclude paths involving only one arc because they naturally fail in providing anonymity. The subproblem FINDSG$(G, \mathcal{F}, P \setminus \{b\}, \{(i,j)\})$ requires, in this case, to find two paths: one from s to i and one from j to t. This can be done in $O(n+m)$ using a graph traversing algorithm.

In this case we can also provide a more efficient algorithm. Checking the anonymity property for all pairs of distinct vertices s, $t \in V$ is equivalent to verify the existence of two different elementary paths between every pair of vertices. If the latter condition is not satisfied by a pair of vertices $i, j \in V$, then the family of paths between i and j cannot be anonymous. Vice-versa, if for all $i, j \in V$ there exist two paths, it is always possible to substitute the arc (i,j) with another path from i to j and thus the anonymity condition is satisfied for every pair of vertices. The computational complexity of verifying the existence of two paths for all pairs of nodes is $O(n^2(n+m))$, which is less than $O(n^2m^3(n+m))$ required by n^2 executions of Alg. 1.

The plot in Fig. 7 reports the result obtained on randomly generated graphs, with the same parameters used for the secret Santa problem. Again the strong anonymity path property becomes more common as the number of vertices and the density of the graph increase.

Anonymous routing protocols usually generate pseudo-random paths in order to maximize the level of anonymity provided and the robustness against traffic analysis attacks. This introduces delays in the transaction (e.g. in onion routing we have to apply a layer of cryptography for each node in the path) that cannot be tolerated in certain applications, i.e. when the content of the message is part of an audio or video stream, or in financial market transactions. In these situations we may want to give up some anonymity in exchange for performances. We may, for example, force the routing protocol to choose paths whose length is close to the shortest path,

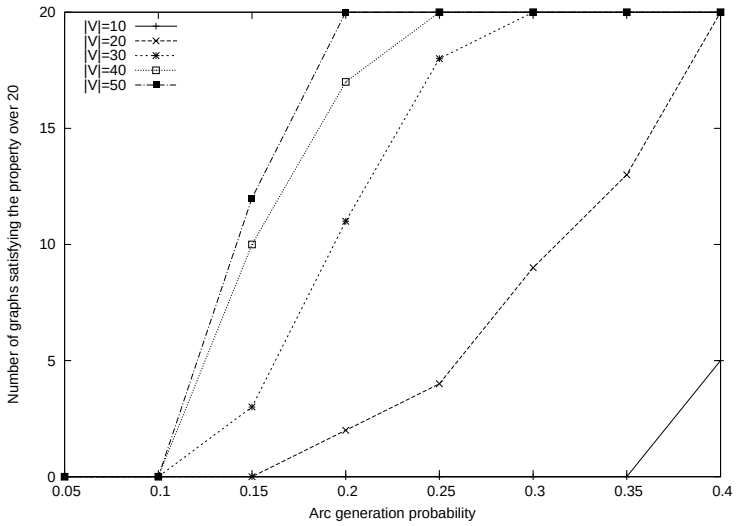

Figure 7: Proportion of random graphs with p ranging in $[0.05, 0.4]$ which do not contain forced paths.

instead of random ones. Ideally, we would like the topology of the graph to allow them to be strongly anonymous in the sense of Defn. 7. We can check this property in a similar way, but this time the family $\mathcal{F}$ will contain only the s-t paths whose length is not greater than α times the length of the shortest path from s to t, where $\alpha \geq 1$ is a given parameter. We have tested this scenario on randomly generated graphs, with 10 or 20 vertices and arc generation probability $p = 0.4$. The plot in Fig. 8 shows, for different values of α, the number of graphs out of 10 that satisfy the strong anonymity path property.

6. Robust path

In this section we describe a third application that makes use of the general version of the ASP (see Defn. 2) to verify that paths in a graph are "robust". Intuitively, given a graph, a path from s to t is said to be robust if from any vertex touched by the path it is always possible to reach t, no matter which arc of the graph becomes unavailable. This is particularly useful in defining routes for emergency services: suppose one needs to travel from a location A to another location B and wants to be sure that in case a street becomes unavailable (e.g. because of a traffic accident) he will not

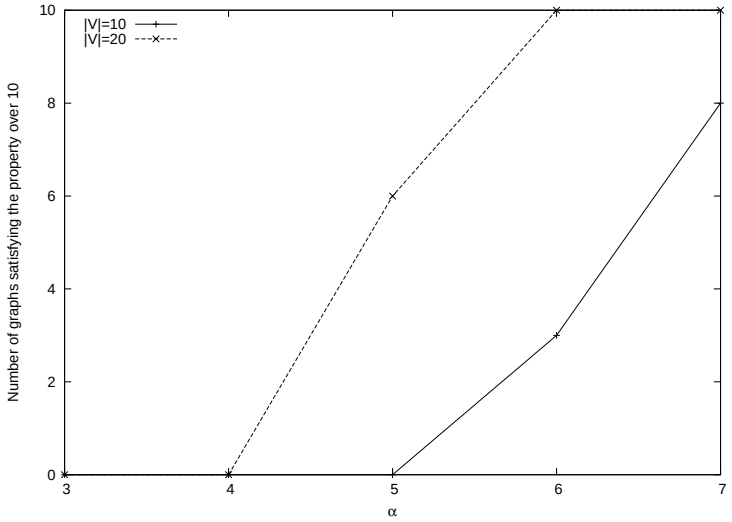

Figure 8: Proportion of random graphs with $p = 0.4$ and 10 or 20 vertices which do not contain forced paths for values of α between 3 and 7.

be stuck. Formally,

Definition 8 (Robust paths). Given a digraph $G = (V, A)$, G has the *robust path property* if, for every path P between two nodes $s, t \in V$ and for all edges $a \in P$, there exists another path P' between s and t s.t. P and P' have the same prefix up to edge a, and a *does not* appear in P'.

We can model the corresponding decision problem as an ASP where $\mathcal{F}$ is the family of all paths between 2 fixed nodes s and t and, given a path S, we consider any initial subpath of length at most $|S| - 1$ a partial view of S. For example, if $S = \{a, b, c, d\}$ (see Fig. 9) it has 4 partial views: $\emptyset, \{a\}, \{a, b\}, \{a, b, c\}$.

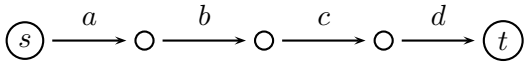

Figure 9: A path and its partial views. A path $S = \{a, b, c, d\}$ has 4 partial views: $\emptyset, \{a\}, \{a, b\}, \{a, b, c\}$.

Consider the network in Fig. 10. The path $\{s, 4, 3, t\}$ is not robust: if arc $(4, 3)$ becomes unavailable while one is traversing arc $(s, 4)$, there is no way to complete the path. From the point of view of anonymity, the partial view

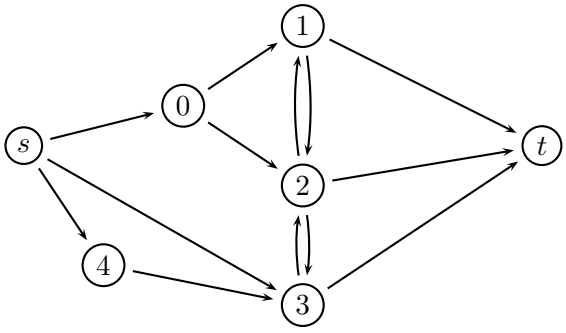

Figure 10: Example of robust, but not anonymous network

$\{(s, 4)\}$ implies that the selected subgraph contains also the arc $(4, 3)$. The path $\{s, 0, 2, t\}$, on the contrary allows at every step to switch to a different path in case of failure of one of the arcs.

This case of the ASP can be solved particularly efficiently. First we observe that for each partial view the last arc is the only one that matters in order to obtain information on the unknown subpath. Moreover the only condition for a vertex $v \in V$ to be safe is the existence of paths that use different outgoing arcs of the vertex v itself.

We propose Algorithm 2 to solve this problem. It incrementally discards unsafe vertices until it obtains a graph $G' = (V', A')$, with $V' \subseteq V$ and $A' \subseteq A$, such that the family of $s - t$ paths in G' is anonymous. Going back to the network in Fig. 10, Alg. 2 removes the unsafe vertex 4 and then returns an affirmative answer. In fact, any $s - t$ path not visiting the vertex 4 is robust.

Theorem 2. *Alg. 2 correctly solves the robust path decision problem.*

Proof. Let $G' = (V', A')$ be the restricted graph obtained at the end of Alg. 2. If the algorithm returns a non empty path, for all vertices $v \in V'$ there exist two paths from v to t which use different outgoing arcs form v. Hence, any initial sub-path ending in a vertex $v \in V$ does not allow to predict any following arc of the path. Thus, the (ordered) set of all the $s - t$ paths visiting only vertices in V' is an anonymous family of subgraphs and all of them are robust. Assume now there is a solution and Alg. 2 returns $\emptyset$. The existence of a solution implies the existence of a non empty anonymous set $\mathcal{Y}$ of paths between s and t. Since at line 16 FindPath(V, A, s, t) returned

$\emptyset$ for every path $S \in \mathcal{Y}$ at least one visited vertex has been removed from V. Consider the first time a vertex v, visited by at least one path $S \in \mathcal{Y}$ was removed from V. Since S is an anonymous path, $\mathcal{Y}$ must contain another path T sharing with S the initial sub-path ending at vertex v, but leaving v through a different arc. Thus the algorithm cannot remove the vertex. Finally, since at each recursive call the cardinality of V decreases by one, the number of recursive calls is bounded by the number of vertices of the original graph. □

The subroutine FindPath requires a visit of the graph, hence its computational complexity is $O(m)$. The number of recursion levels is bounded by the number of vertices and in each level FindPath is executed at most two times for each vertex. Thus the overall complexity of Alg. 2 is $O(n^2m)$.

The algorithm has been tested on two real-world instances. The first graph is a large portion of the directed road network of the city of Rome (Italy). It contains 3353 vertices and 8870 edges. The second one is the directed road network of the city of Milan (Italy) and contains 12442 vertices and 26373 edges. We randomly selected 100 pairs of vertices and run the algorithm to verify the existence of a family of robust paths. In the graph of Rome we obtained a positive result in 70% of the cases. The length of the shortest robust path is on average only 5% greater than the shortest path and in 30 cases over 100 the shortest path is a robust path. The same test gave quite different results on the graph of Milan: a family of robust paths exists only for 5 pairs out of 100 and the average increase of length with respect to the shortest path is 46%. The reason for this wide discrepancy is probably a different level of detail in the description of the road network. Indeed, the way crossroads, traffic circles and slip roads are mapped into the graph can deeply influence the existence of a robust path.

7. Conclusions

In this paper we studied a notion of anonymous subgraph and anonymous family of subgraphs. We formally defined the Anonymous Subgraph Problem that, given a directed graph, a family of subgraphs and a partial view function, asks to decide if the given family of subgraphs contains an anonymous one. We described a restriction of the problem, when only one arc of the subgraph is known, and proposed an algorithm to solve it. We studied the condition for the algorithm to be polynomial in the size of the graph. We described examples of applications from different fields that can be modelled as ASP: the first concerning the anonymous exchange of

Algorithm 2 Algorithm for solving the ASP for robust path on graph $G = (V, A)$. FindPath(V, A, v, t) returns a path $S \subseteq A$ from v to t in graph (V, A). It returns $\emptyset$ if such path does not exist. S_i denotes the i-th arc in path S.

```
1: FindRP(V, A, s, t)
2: for all v ∈ V do
3:    S = FindPath(V, A, v, t)
4:    if S = ∅ then
5:       V = V \ {v}
6:       return FindRP(V, A, s, t)
7:    end if
8:    A = A \ {S_1}
9:    if FindPath(V, A, v, t) = ∅ then
10:      V = V \ {v}
11:      return FindRP(V, A, s, t)
12:   else
13:      A = A ∪ {S_1}
14:   end if
15: end for
16: return FindPath(V, A, s, t)
```

gifts among a group of people, the second related to anonymous routing in telecommunication networks and the last one in the field of transportation. For each of these we have provided a detailed experimental evaluation using both randomly generated and real-life instances of graphs.

[1] L. Backstrom, C. Dwork, J. Kleinberg, Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography, in: WWW '07: Proceedings of the 16th international conference on World Wide Web, ACM Press, New York, NY, USA, 2007, pp. 181–190.
URL http://dx.doi.org/10.1145/1242572.1242598

[2] M. Hay, G. Miklau, D. Jensen, S. Srivastava, Anonymizing social networks, Tech. rep., University of Massachusetts Amherst (2007).

[3] B. Zhou, J. Pei, Preserving privacy in social networks against neighborhood attacks, in: Proceedings of the 24th IEEE International Conference on Data Engineering (ICDE'08), IEEE Computer Society, Cancun, Mexico, 2008, pp. 506–515.

[4] E. Zheleva, L. Getoor, Preserving the privacy of sensitive relationships in graph data, in: Privacy, Security, and Trust in KDD, Vol. 4890, Springer, 2008, pp. 153–171.

[5] K. B. Frikken, P. Golle, Private social network analysis: How to assemble pieces of a graph privately, in: Proceedings of the 5th ACM Workshop on Privacy in Electronic Society (WPES), 2006, pp. 89–98.

[6] T. Feder, S. U. Nabar, E. Terzi, Anonymizing graphs, CoRR abs/0810.5578.

[7] D. Chaum, The dining cryptographers problem: Unconditional sender and recipient untraceability, Journal of Cryptology 1 (1988) 65–75.

[8] A. Serjantov, G. Danezis, Towards an information theoretic metric for anonymity, in: Privacy Enhancing Technologies, Vol. 2482, Springer-Verlag, 2002, pp. 259–263.

[9] C. Diaz, S. Seys, J. Claessens, B. Preneel, Towards measuring anonymity, in: Privacy Enhancing Technologies, Vol. 2482, Springer-Verlag, 2002, pp. 54–68.

[10] G. Tóth, Z. Hornák, Measuring anonymity in a non-adaptive, real-time system, in: Proceedings of Privacy Enhancing Technologies workshop (PET 2004), Vol. 3424, Springer-Verlag, 2004, pp. 226–241.

[11] R. E. Newman, I. S. Moskowitz, P. Syverson, A. Serjantov, Metrics for traffic analysis prevention, in: Proceedings of Privacy Enhancing Technologies Workshop (PET 2003), Springer-Verlag, LNCS, 2003, pp. 48–65.

[12] M. Edman, F. Sivrikaya, B. Yener, A combinatorial approach to measuring anonymity, in: Intelligence and Security Informatics, 2007 IEEE, 2007, pp. 356–363.
URL http://dx.doi.org/10.1109/ISI.2007.379497

[13] B. Gierlichs, C. Troncoso, C. Diaz, B. Preneel, I. Verbauwhede, Revisiting a combinatorial approach toward measuring anonymity, in: WPES '08: Proceedings of the 7th ACM workshop on Privacy in the electronic society, ACM, New York, NY, USA, 2008, pp. 111–116. doi:http://doi.acm.org/10.1145/1456403.1456422.

[14] P. Meurdesoif, P. Pesneau, F. Vanderbeck, Meter installation for monitoring network traffic, in: International Network Optimization Conference (INOC), 2007.

[15] R. Fagin, J. Y. Halpern, Y. Moses, M. Y. Vardi, Reasoning About Knowledge, The MIT Press, 1995.

[16] L. Liberti, F. Raimondi, The secret santa problem, in: R. Fleischer, J. Xu (Eds.), AAIM08 Proceedings, Springer, 2008, pp. 271–279.

[17] J. Park, M. E. J. Newman, Origin of degree correlations in the internet and other networks, Phys. Rev. E 68 (2) (2003) 026112.

[18] D. Chaum, Untraceable electronic mail, return addresses, and digital pseudonyms, Communications of the ACM 24 (1981) 84–88.

[19] P. F. Syverson, D. M. Goldschlag, M. G. Reed, Anonymous connections and onion routing, in: Proceedings of the 1997 IEEE Symposium on Security and Privacy, 1997, pp. 44–54.

[20] M. Reiter, A. Rubin., Crowds: anonymity for web transactions, ACM Transactions on Information and System Security 1(1) (1998) 66–92.