The Anonymous Subgraph Problem

Andrea Bettinelli

Dept. of Mathematics, Universitá degli Studi di Milano, Italy

Leo Liberti

LIX, École Polytechnique, F-91128 Palaiseau, France

Franco Raimondi

Dept. of Computer Science, University College London, UK

David Savourey

LIX, École Polytechnique, F-91128 Palaiseau, France

Key words: anonymity, anonymous routing, secret santa, graph's topology

1 Introduction

Many problems can be modeled as the search for a subgraph $S \subseteq A$ with specific properties, given a graph $G = (V, A)$. There are applications in which it is desirable to ensure also S to be *anonymous*. 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 details several specific family of subgraphs.

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

2 Characterization of anonymity

Given a digraph $G = (V, A)$, let $|V| = n$ and $|A| = m$. We are interested in finding if a certain family $\mathcal{A}$ of subgraphs is *anonymous* with respect to G. By anonymous we mean that it is not possible to single out a subgraph $S \in \mathcal{A}$, nor to identify any other arc in the subgraph, given the topology of the graph and a subset C of the arcs in S. We call C a *partial view* of S. Let $PV : \mathscr{P}(A) \times \mathcal{A} \to \{0, 1\}$ be the funcion

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

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

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

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

We call *anonymous subgraphs* the elements of an anonymous family $\mathcal{A}$. It is now possible to define *Anonymous Subgraph Problem* (ASP) as the decision problem of checking if a family of subgraphs contains an anonymous family with respect to a graph.

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

Here we restrict our analysis to the case where the set of partial views of a subgraph S is $\{C | \, C \subseteq S \wedge |C| = 1\}$, i.e. only one arc of the subgraph is known. With this restriction, we obtain the following definition of anonymity:

Definition 3 *Given a digraph $G = (V, A)$, a family of subgraphs $\mathcal{A} \subseteq \mathscr{P}(A)$ is* anonymous *in G if*

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

We will refer to ASP1 to denote the Anonymous Subgraph Problem where Definition 3 is used to characterize anonymity. In the next section we propose an algorithm to solve the ASP1 and we show under what conditions its computational complexity is polynomial in the size of the graph G, even if the family $\mathcal{S}$ contains a combinatorial number of subgraphs.

1: FindAnonymousSG$(G, \mathcal{S}, P)$:
2: **for all** $a \neq b \in P$ **do**
3: **if** FindSG$(G, \mathcal{S}, P \setminus \{b\}, \{a\}) = \emptyset$ **then**
4: **return** FindAnonymousSG$(G, \mathcal{S}, P \setminus \{a\})$
5: **end if**
6: **end for**
7: **return** FindSG$(G, \mathcal{S}, P, \emptyset)$

Algorithm 1. Algorithm for solving the ASP1

3 Algorithm

The algorithm 1 solves the ASP1: it returns an element of $\mathcal{A}$, if $\mathcal{A}$ exists, and an empty set otherwise. It is a recursive algorithm and at the top level P is equal to the arc set A. The algorithm is based on the following observation: if there exists two distinct arcs $a, b \in A$ such that no subset $T \in \mathcal{S}$ contains a but not b, it implies that all the subsets $S \in \mathcal{S}$ that contain a are not anonymous. Thus, we can transfer the anonymity property from the subsets to the arcs. The algorithm iteratively remove arcs from the set P of permitted arcs and uses this set as additional constraints when looking for possible subgraphs. If no subgraphs can be found satisfying the additional constraints given by P the family is not anonymous in G.

We assume the correctness of the subroutine FindSG$(G, \mathcal{S}, P, X)$: it returns an empty set if and only if it doesn't exist a subgraph $S \in \mathcal{S} \cap \mathscr{P}(P) : \ x \in S \forall x \in X$.

Theorem 4 *Alg. 1 correctly solves the ASP1.*

Proof. First we observe that, if the algorithm returns a non empty solution, at line 7 the set $\mathcal{A}$ of subgraphs in $\mathcal{S}$ where every arc belongs to P is anonymous in G. Let $S \subseteq P$ be an element of $\mathcal{A}$, we know that $\forall a, b \in S \ \exists T \in \mathcal{A}$ s.t. $a \in T$ and $b \notin T$, otherwise a would have been banned from P. Assume now there is a solution to ASP1 and Alg. 1 fails. The existence of a solution implies the existence of a non empty anonymous set $\mathcal{A}$. If our algorithm reached line 7 with $\mathcal{A} \subseteq \mathscr{P}(P)$, then, because FindSG$(G, \mathcal{S}, P, X)$ is correct, our algorithm would not have failed. Thus we know that the algorithm reached line 7 with $\mathcal{A} \setminus \mathscr{P}(P) \neq \emptyset$. Consider the first time an arc $a \in A$ used in at least one element of $\mathcal{A}$ has been removed from P. At that time $\mathcal{A} \subseteq \mathscr{P}(P)$, so a would not have been removed because $\forall b \neq a \in P \ \exists T \in \mathcal{A}$ s.t. $a \in T$ and $b \notin T$.

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 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 to solve the subproblem, we can solve ASP1 in polynomial time.

4 Special cases and applications.

Definition 1 holds for a generic family of subsets. In real application we usually have to deal with a family $\mathcal{S}$ characterized by specific properties. By exploiting them, we can describe $\mathcal{S}$ implicitly and, in some cases, obtain polynomial procedures to solve FINDSG even if the cardinality of $\mathcal{S}$ is combinatorial in the size of the graph.

We now analize some families of subgraphs that lead to interesting applications.

Secret Santa Problem.

If the family $\mathcal{S}$ is the set of all *Vertex Disjoint Circuit Covers* (VDCCs), we obtain the Secret Santa Problem described in [2].

The basic concept of the Secret Santa game is simple. All of the participants' names are placed into a hat. Each person then chooses one name from the box, but doesn't tell anyone which name was picked. He/she is now responsible for buying a gift for the person selected. When the Secret Santa wraps his/her gift, he/she should label it with the recipient's name but doesn't indicate whom the present is from. 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 pairs of participants should be avoided. The problem can be modeled with a digraph, where vertices represent the participants and arcs the possibility of a participant giving a gift to another participant. We want to determine if the topology of the graph allows an anonymous exchange of gifts, that is nobody can discover who made a gift to whom, knowing the graph and the receiver of his gift.

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

Definition 5 *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.

In this case FINDSG$(G, \mathcal{S}, P, \{(i, j)\}$ requires to find a VDCC with restrictions on the arcs can be used. As shown in [2] 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 $A' = P$.

Anonymous routing.

In many contexts it is desirable to hide the identity of the users involved in a transaction on a public telecommunication network. According to the specific application, we may be interested in:

- *sender anonymity* to a node, to the receiver or to a global attacker;
- *receiver anonymity* to any node, to the sender or to a global attacker;
- *sender-receiver unlinkability* to any node or a global attacker. This means that a node may know that A sent a message and B received one, but not that A's message was actually received by B.

Several protocols to provide anonymous routing features has been proposed in the literature ([1,4,3]). Using these protocols every node traversed by a message has only a partial knowledge on the path the message is being routed on. Typically a node knows only the next step, or the next and the previous one.

Attacks against these protocols are usually based on traffic analysis. Thus, if the topology of the network contains "forced paths", they can leak information to an attacker who is monitoring the traffic.

Some protocols, like Onion Routing, require the topology of the network to be known to every participant. Every time a node leave or a new node joins the protocol, the topology of the network changes. Therefore it may be useful to check the network against the presence of "forced paths". This can be done by solving, for each pair of nodes (s, t) of the network, an instance of ASP1 where G is the graph representing the network and S is the family of all paths of length at least 2 between the two nodes. We exclude paths involving one arc because they naturally fail in providing anonymity. The subproblem FINDSG$(G, \mathcal{S}, P \setminus \{b\}, \{(i, j)\})$ requires, in this case, to find two paths: one from s to i and one from j to t. It can be done in $O(n + m)$ using a graph traversing algorithm.

Anonymous routing protocols usually generate pseudo-random path in order to maximize the level of anonymity provided and the robustness against traf-

fic 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 application, 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 *quasi* shortest paths, instead of random ones. Again, we would like the topology of the graph to allow them to be anonymous. We can check this property in a way similar to what we have done in the previous case, but this time the family $\mathcal{S}$ will contain only the $s-t$ paths S 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.

References

[1] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24:84–88, 1981.

[2] Leo Liberti and Franco Raimondi. The secret santa problem. In R. Fleischer and J. Xu, editors, *AAIM08 Proceedings*, pages 271–279. Springer, 2008.

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

[4] P F Syverson, D M Goldschlag, and M G Reed. Anonymous connections and onion routing. *In IEEE Symposium on Security and Privacy*, 1997.