Introducing ADegree: anonymisation of Social Networks through Constraint Programming

Sergei Solonets, Victor Drobny, Victor Rivera, and JooYoung Lee

Innopolis University, Innopolis, Russia,
Institute of Technologies and Software Development
{s.solonets, v.drobnyy, v.rivera, j.lee}@innopolis.ru,

Abstract. With the rapid growth of Online Social Networks (OSNs) and the information involved in them, research studies concerning OSNs, as well as the foundation of businesses, have become popular. Privacy on OSNs is typically protected by anonymisation methods. Current methods are not sufficient to ensure privacy and they impose restrictions on the network making it not suitable for research studies. This paper introduces an approach to find an optimal anonymous graph under user-defined metrics using Constraint Programming, a technique that provides well-tested and optimised engine for combinatorial problems. The approach finds a good trade-off between protection of sensitive data and quality of the information represented by the network.

Keywords: Anonymisation; Online Social Networks; Constraint Programming;

1 Introduction

Social networks (SNs) are social structures made up of individuals (or organisations) that are connected by one or more types of interdependency. Individuals are called "nodes" and connections are called "edges". There are several types of SNs; for example, in LinkedIn[1] (an online network of professionals) every link between two users specifies a professional relationship between them [10]. In Facebook[2] or VK[3] links correspond to friendship. Each SN shares information according to the type of the network. Given the rapid growing of SN and the information involved in the networks, several research studies have been made [7, 9, 11, 12, 14, 16].

The information carried out by Social Networks is of paramount importance in domains such as marketing [8]. Network owners often share this information with advertising partners and other third parties. Such practice is the foundation of the business case for many social network owners, that have made the analysis of social networks profitable. However, these kind of marketing makes the owners of social networks like Facebook and Twitter (http://www. twitter.com)

[1] www.linkedin.com
[2] www.facebook.com
[3] www.vk.com

increase the sharing of potentially sensitive information about users and their relationships. Owners are compelled by law to respect the privacy policies of their users. This privacy is typically protected by anonymisation.

Anonymisation methods are used to assure privacy policies of the users of SN (i.e., sensitive information). The common intuitive method is to remove the identity of each node in the graph, replacing it with a random identification number. However, Backstrom et. al. [1] have shown that this method is not adequate for preserving the privacy of nodes. Specifically, the authors show that in such an anonymised network, there exists an adversary who can identify target individuals and the link structure between them.

There have been some other attempts to come up with more advanced anonymisation approaches. As an example, Liu and Terzi [13] consider node re-identification assuming that the adversaries' auxiliary information consists only of node degrees. Campan and Truta [3] propose metrics for the information loss caused by edge addition and deletion and apply k-anonymity to node attributes as well as neighbourhood structure, among others [6, 20].

Despite the fact that there exist many anonymisation methods, those concerning to remove private information of nodes and those that change the structure of the the network (e.g., based on k-degree anonymity) are not sufficient for privacy when dealing with social networks [1,15]: the fundamental issue when removing private information of nodes (e.g. names, addresses) is that even though there are millions of people participating in a social network, each node has a relatively unique relationship with his/her neighbours. This uniqueness can be exploited in order to identify the participants in the network; the fundamental issue when changing the structure of the graph (e.g. based on k-anonymity) is that they impose arbitrary restrictions on the network and make arbitrary assumptions about the properties of the given graph [15].

We consider a scenario where the owner of a social network wants to release the underlying SN graph preserving the privacy of its users. This paper introduces the formal definition of an anonymisation approach using Constraint Programming (CP) in its implementation. Our approach finds out a network that respects the privacy polices, hides the private information, makes it harder to de-anonymise the network and, as opposed to current anonymisation approaches, conserves the properties of the network as much as possible. On the other hand, [13] only considers the degree attack scenario. Due to the nature of information carried out by the Social Networks, only k-automorphism constraint can guarantee privacy under any type of attack [21]. However this kind of constraint is very heavy and as a consequence leads to a huge information loss. Therefore, to find a network as close as possible to the original, we need to study possible scenario attacks in every particular case.

2 Preliminaries

2.1 Constraint Programming

Constraint programming (CP) [17] is a paradigm for solving combinatorial search problems that draws on a wide range of techniques. CP is currently applied with success to many domains, such as scheduling, planning, vehicle routing, configuration, networks, and bioinformatics. The focus of CP is on reducing the search space by pruning values that cannot appear in any feasible or optimal solution. Constraints are relations, and a constraint satisfaction problem (CSP) states which relations should hold among the given decision variables. Constraint solvers take a real-world problem and represent in terms of decision variables and constraints to find an assignment to all the variables that satisfies the constraints. To get a solution of a CSP, one uses the following generic algorithm:

1. Use propagation to prune the domains of the variables. This step is executed by propagators. Propagators are processes that filter the domains of a set of finite domain variables according to the semantics of the constraint they implement (also called pruning rules). Propagators share a common store which contains the information that is currently known about the variables of the problem. As soon as a propagator is able to infer new information from the store, it adds this new information in the store. This iterative steps stop when the propagator(s) reaches a fixed point, i.e., when it cannot prune more values.
2. When the propagators reach a fix point, we may have three possible situations:
   - all the variables are bound to a value. In this case the search stops since a solution has been obtained.
   - there are some variables that are yet to be determined. In this case, we split the CSP into new problems according to branch strategies, generating new constraints. Then the step 1 is triggered, we wait until propagators get stable to continue.
   - there is at least one variable whose domain is empty. In this case, we go back to the previous step.

2.2 Social Networks as Graphs

A social network can be seen as a social structure represented by a graph where nodes represent actors of the social network and edges represent a interdependency among those actors. Edges can represent specific types of interdependency, such as friendship, kinship, common interest, financial exchange, dislike, and so forth. The following is the definition of a Social Network which will be used throughout this paper.

Definition 1 (Social Network). *A social network SN is a directed/undirected graph* $G = (V, E)$*, containing*

1. *a set of attributes for each node in V and*
2. *a set of attributes for each edge in E.*

Typically, Social Network analysers are interested in sub-populations of social graphs or sub-links of the graphs. For instance, demographic studies might be interested in relationships between people of the same gender in ages of a specific range. This categorization of graphs is represented by Restricted Graphs:

Definition 2. ***Restricted Graph:*** *Let $G(V, E)$ be a graph. $G^-\langle V^-, E^-\rangle$ is a restricted graph with $V^- \subseteq V$ and $E^- \subseteq E$*

An interesting property of data anonymisation is k-anonymity introduced in [19]. k-anonymity solves the problem: "Given person-specific field-structured data, produce a release of the data with scientific guarantees that the individuals who are the subjects of the data cannot be re-identified while the data remain practically useful." Liu et al. [13] introduced the concept of k-degree anonymous graphs. The following is the formal definition as used in the paper. We first define a degree sequence for the graph.

Proposition 1. ***Degree sequence:*** *Let dG be sequence. dG is a degree sequence of graph $G(V, E)$ iff is a sequence of size $n = |V|$ such that $dG[i]$ is the degree of the i-th node of G.*

The definition of k-anonymity for the degree sequence is as follows.

Proposition 2. ***k-degree anonymous:*** *A degree sequence dG is k-anonymous, if every distinct value in dG appears at least $k - 1$ times. That is*

$$\forall i \in dG \Rightarrow occur(i, dG) \geq k,$$

where $occur(v, s)$ is the number of occurrences of v in sequence s.

Finally, we state that

Definition 3. *a k-degree anonymous graph has associated a k-degree anonymous sequence.*

3 Related Work

Privacy on social networks is typically protected by anonymisation methods. An initial anonymisation attempt was to remove sensitive information of the nodes, such as names, addresses and phone numbers. However, this kind of method is easy to de-anonymise by studying the structure of the network. There have been several anonymisation methods that modify the initial structure of graphs that represent Social Networks to avoid users being uncovered by analysing the structure of their connections. Liu and Terzi anonymise the network by making it k-degree anonymous [13]. A network is k-degree anonymous if for every node

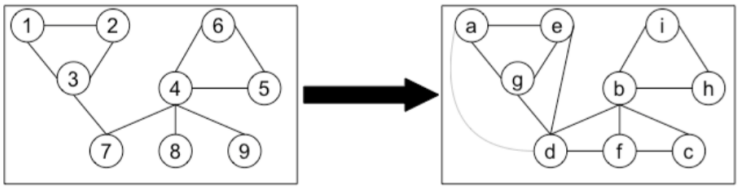

Fig. 1. Liu et al. approach.

v, there exist at least $k-1$ other nodes in the network with the same degree as v. Figure 1 shows an example of such anonymisation.

Campan and Truta suggest anonymising the network by applying the concept of edge generalizing on the corresponding graph [3]. Edge generalisation is implemented by clustering nodes. Every cluster is replaced with a new node whose neighbours are the union of the neighbours of the nodes in the cluster. Each new node is associated with a pair of integers (#n,#e) representing the number of nodes and the number of edges inside the cluster. These pairs of integers are then used to approximate the interconnection of those nodes inside the cluster without revealing information about how two nodes in separate clusters are interconnected. Figure 2 shows an example of such anonymisation. Notice that in each case the interconnection of the nodes inside every cluster can be inferred. For instances, there is only one instance of connecting 3 nodes with 3 edges. However, the connections between nodes of different clusters are hidden by the anonymisation process.

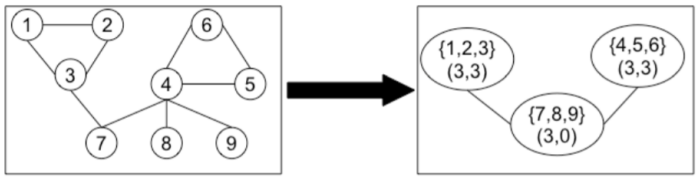

Fig. 2. Campan et al. approach.

Bhagat et al anonymise the network by associating each node with a list of possible *ids* [2]. Each list contains the real *id* of the node. The sensitive information is therefore protected, since, in general, it is not possible to infer whether a node is present in the network and whether there is a connection between two nodes. Bhagat et al. shows, however, that these lists of *ids* need to be carefully computed since it is not impossible to unmask individuals and connections. Fig-

ure 3 shows an example of such anonymisation. Notice that in every case the *id* of the node has been replaced with a list of possible *ids* containing the real one.

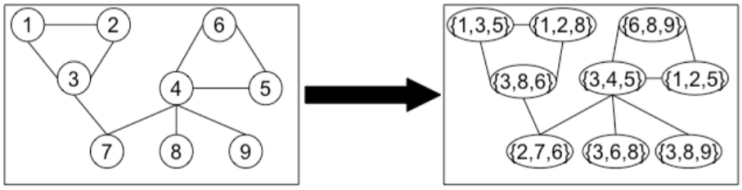

Fig. 3. Bhagat et al. approach.

4 Anonymising Social Network: `ADegree`

`ADegree` is a constraint that is used to find an anonymised network that protects users' sensitive information by ensuring k-degree anonymity, making the reverse process (de-anonymisation) harder to achieve. The anonymised network preserves the structure of the original network as much as possible. This structure is then preserved by defining the interest of the Social Network's analyser. Typically, Social Networks analysers are interested in subgraphs within the Social Network. Section 4.1 defines a way to represent such interest and Section 4.2 formally defines the constraint for the anonymisation of Social Networks.

4.1 Query of Interest

The fundamental problem when altering the structure of a graph is that arbitrary restrictions are imposed on the network as well as arbitrary assumptions are made about its properties. The problem is due to the ignorance on the utility of anonymised graph. Companies dedicated to the study of social networks have specific queries of interest. For example, 'how many users are there in some specific sub-populations?', 'What are the patterns of interaction and friendships?', 'Which sub-populations are interacting?'. These kind of queries define the utility of the network and can be used to determine how close an anonymised graph is from the original one.

The following formalises the query of interest.

Definition 4. ***Query of Interest:*** *A query of interest QI is a tuple $\langle G, P, T, F\rangle$ where*

$G(V, E)$ is a graph (representing a Social Network);
$P \subseteq \mathbb{P}(V)$ is set of set of nodes;
$T \subseteq \mathbb{P}(label(E))$ is set of set of labels.;
$F \in \{G^{-}\langle p, t\rangle \mid p \in P \wedge t \in T\} \rightarrow \mathbb{Z}$
where $\mathbb{P}$ is the power set of a set.

In Definition 4, P defines the different categories of 'users' (actors) that the query is interested in. T defines the different categories of 'labels' that the query is interested in. As an example, take the phone communication company AT&T social network that represent people as nodes, phone calls between people as edges and the duration of calls as edges' labels. In a possible QI, P categorises the set of user the company is interested in, e.g. teenagers and females over 50 years, and T categorises the set of labels, e.g. 3 to 15 minutes. F is a function (defined by the user) that represents the closeness of G^- to G.

4.2 `ADegree`

The constraint is defined as follows.

Definition 5. *`ADegree`: Suppose $G(V, E)$ and $G'(V, E')$ are graphs representing social networks. Let $QI = \langle G, P, T, F \rangle$ be the query of interest, and let k and v be integers. The constraint `ADegree` holds iff*

1. *G' is k-degree anonymous;*
2. *G' is as close as possible to G*

$$\sum_{p \in P, t \in T} abs(F(G, p, t) - F(G', p, t)) \leq v;$$

   where abs is the absolute value
3. *if G' is already k-degree anonymous, then $G' \neq G$*

The solution, G', is a graph representing a social network that

- is an anonymised network with k-degree anonymity property that
  - hides user sensitive information;
  - makes it more difficult to de-anonymise the network.
- is as close as possible to the original which implies that the information has fewer changes. Hence, any study performed on it will be significant.

5 Implementation of `ADegree`

We implemented `ADegree` in Gecode [5]. The implementation models the problem as a Constraint Satisfaction Problem (CSP). It finds an anonymised graph from a given social network. The found graph protects sensitive information whilst preserving the structure of the graph as much as possible. Gecode is a powerful tool for solving CSPs. The idea behind Constraint Programming is to define the model as a set of constraints over finite domain variables. Gecode framework will prune variables' domain until it finds a solution. A solution is found whenever the domain of all variables in the model has been reduced to a value. A user defines how to prune variables' values by specifying branching rules.

Part of previously described formal definitions are implemented in this study since our current implementation does not take into account labelled graphs. Subsection 5.2 is devoted to explain the possible implementations. Current implementation of `ADegree` can be found in [18].

5.1 ADegree

When working with Constraint Programming, one needs to define the followings.

- finite domain variables;
- constraints to be satisfied by the value of the variables;
- branching rules and
- the search engine.

Finite domain variables: `ADegree` is a data structure that represents a k-degree anonymous undirected-graph. `ADegree` is represented by a set of adjacent nodes.

Constraints: The set of constraints that needs to be satisfied is that the resulting graph is

cons1 : a super-graph of the input graph and
cons2 : k-degree anonymous,
cons3 : as close as possible to the input graph.

In order to ensure that **cons1** holds it is necessary to add the following constraint.

- For every edge that is represented by pair of nodes (a, b), a is in an adjacent set of b and b is in an adjacent set of a.

In order to ensure that **cons2** holds we need to know that for every node there is at least $k-1$ nodes with the same amount of adjacent nodes. We introduced a degree sequence variable and constraints are as following.

- The size of the degree sequence is the number of nodes in the graph.
- Each value in the degree sequence corresponds to the cardinality of the corresponding adjacent set.
- For every value of degree there is at least $k-1$ degrees with the same value.

Branches: It is important to define the branch strategy to be able to find a solution in an acceptable time. For instance, choosing any branch strategy on the edges of the graph and running Depth First Search (DFS) engine will yield a solution. However, the search engine will try, for every possible super-graph, to check whether it has a k-degree anonymous sequence or not. This approach is feasible on small graphs but the time complexity will increase significantly as the graph increase its size.

In order to increase performance several methods can be applied. The first technique is to optimise the constraints. Gecode can cut off the search space by defining stronger conditions. One way to define such stronger constraints is to define a sorted degree sequence as realizable. That is, there exists a simple graph whose nodes have precisely the sorted degree sequence. Erdös and Gallai

described the necessary and sufficient condition for a degree sequence to be realizable [4]. These conditions are n inequalities on sorted degree sequence. It can be achieved by using sorted versions of numerical arrays provided by Gecode.

For every $l \in [1, n-1]$

$$\sum_{i=1}^{l} d_i \leq l(l-1) + \sum_{i=l+1}^{n} min(l, d_i)$$

where d_i is i-th element in a sorted degree sequence. $min(l, d_i) \leq d_i$ we can transform it into weaker but faster condition:

$$\sum_{i=1}^{l} d_i \leq l(l-1) + \sum_{i=l+1}^{n} d_i$$

This solution has an implicit advantage. To find the existence of k same degree values in a sorted degree sequence, it is enough to search only in $k-1$ surrounding, not in the whole sequence. Such constraints are much lighter because they are from $2k-1$ variables and not from n.

All previous improvements deal only with degree sequence and with its sorted version but not with edges. Our implementation does not take into account pruning of edges. Introducing edges to the branching strategy will slow down the execution if the branch strategy is not carefully defined. Section 5.2 proposes how to solve the problem. The main problem is that the search engine checks several graphs with the same degree sequence even if they are not k-degree anonymous. Hence, the branch strategy needs to be defined so that it branches first on the degree sequence (most of the constraints make restrictions on a degree sequence) rather than on the edges. This will make the search run faster. Gecode allows users to define any order of search. The following is the ideal searching order:

1. Branch on sorted degree sequence
2. Branch on degree sequence
3. Branch on edges

Search: Gecode provides two search engines: Depth First Search (DFS) and Branch and Bound (BAB). DFS gives a solution which satisfies all the constrains but it cannot compare solutions to provide the best one. BAB is designed to give the best solution. For graph anonymisation, the idea is to find a graph that is as close as possible to the initial input. This gives a cost function that can be used to compare results given by the search engine. The cost function is defined by a user and corresponds to the Query of Interest (QI). Once the QI is defined, BAB gives some solution in increasing (decreasing) order until it achieves the best one.

Depending on the cost function, different strategies of searching within one structure can give different execution times. For example, if the cost function is the sum of the degree sequence and its goal is to minimize the sum (e.g. to take the least possible value of a degree first), the solution can be found faster.

5.2 Towards edge labelling

This section is devoted to express how to improve the implementation in order to consider labelling. The main idea is to transform this new problem into the previous one by defining the degree sequence as a tuple of degrees for every label. We need to be sure that every node appears either 0 or at least k times in the graph. If there are m labels, each degree is represented by $(d_1, d_2, \ldots, d_m)$ where d_i is the degree of i-th label of node. This tuple can be written as $deg = d_1 + d_2K + d_3K^2 + ... + d_mK^{m-1}$. Where K is a large integer that is bigger than maximum possible degree value. For example, if we assume that there can be only one edge with same label from node a to b, then K can be $n-1$ (where n is the length of the sequence). By doing this transformation we uniquely assign an integer number to any possible combination of a degree sequence.

Applying this transformation we can now anonymise undirected graphs with labelled edges. Now we can transform this variation to the directed graph case. The only thing needed is to assign different labels to the beginning of the edge and to its end.

6 Using `ADegree`

Consider the graph $G(V, E)$ depicted in Figure 4. The graph shows an excerpt of a social network representing a AT&T network. Nodes represent users, arcs represent calls made between two users, and labels represent time of the calls.

A typical scenario in which AT&T might be interested is studying the graph to come up with better call plans targeting a group of users. The scenario also assumes that AT&T outsources this study (necessity of anonymised the graph). The query of interest (i.e. $QI = \langle G, P, T, F \rangle$) is to find out what is the average number of calls made by a type of user (e.g., teenagers) during a period of time (e.g., peak hours), in order to make a specific plan of minutes. Users are classified by their ages and the calls are classified by the period of time. Let P be the set of possible groups of ages and T be the set of possible periods of time as shown in Table 1.

Table 1. Classification of the AT&T social network.

Table 2. Set P

Nodes	**Type**
$\{0, 1, 2, 7\}$	p_1
$\{4, 8\}$	p_2
$\{3, 5\}$	p_3
$\{6\}$	p_4

Table 3. Set T

Arcs	**Type**
$\{a\}$	t_1
$\{b\}$	t_2

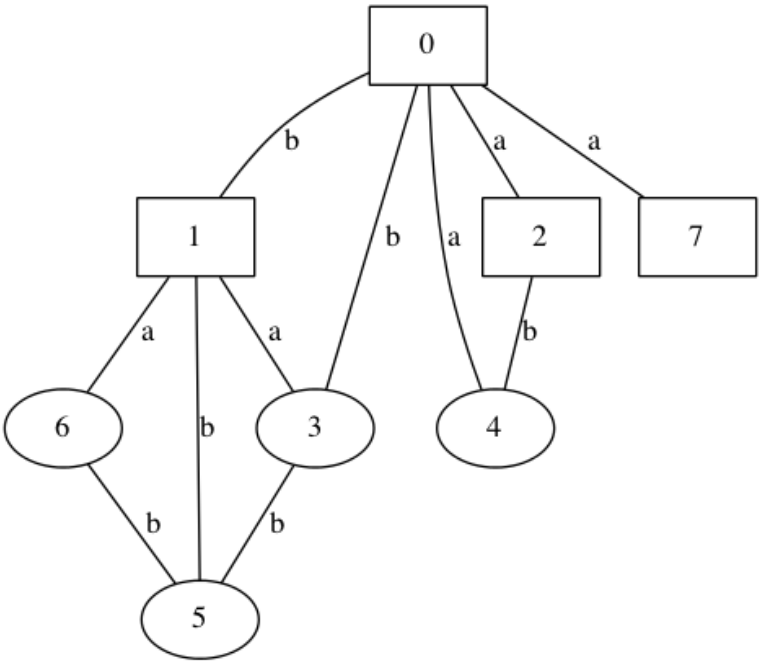

Fig. 4. Modified AT&T network.

Function F is formally defined as, $\forall p, t \cdot p \in P \wedge t \in T \Rightarrow$

$$F(G^-(p,t)) = \frac{\sum_{v_i \in V^-} \left|\{v_2 \mid v_2 \in V^- \wedge (v_1, v_2) \in E^-\}\right|}{|V^-|}$$

where $G^-(p,t) = (V^-, E^-)$ is the restricted graph obtained from G when restricting the set of nodes to those of classification p and the set of arcs to those associated with classification t. An example of the weight associated to the graph taking the type of user teenagers (p_1) during a period of time 5 minutes (t_1) is $F(G^-_{p_1,t_1}) = 0.5$.

The anonymised Social Network found by `ADegree` is depicted in Figure 5. The graph is a 2-degree anonymous. Dotted edges were added by our solution where shapes of nodes denote equivalence classes.

7 Conclusions

The use of social networks has increased exponentially and so as the information shared by owners of these networks to different entities. This information is of special interest for many companies to conduct studies such as marketing or epidemiology. The owners of the networks are compelled by law to protect

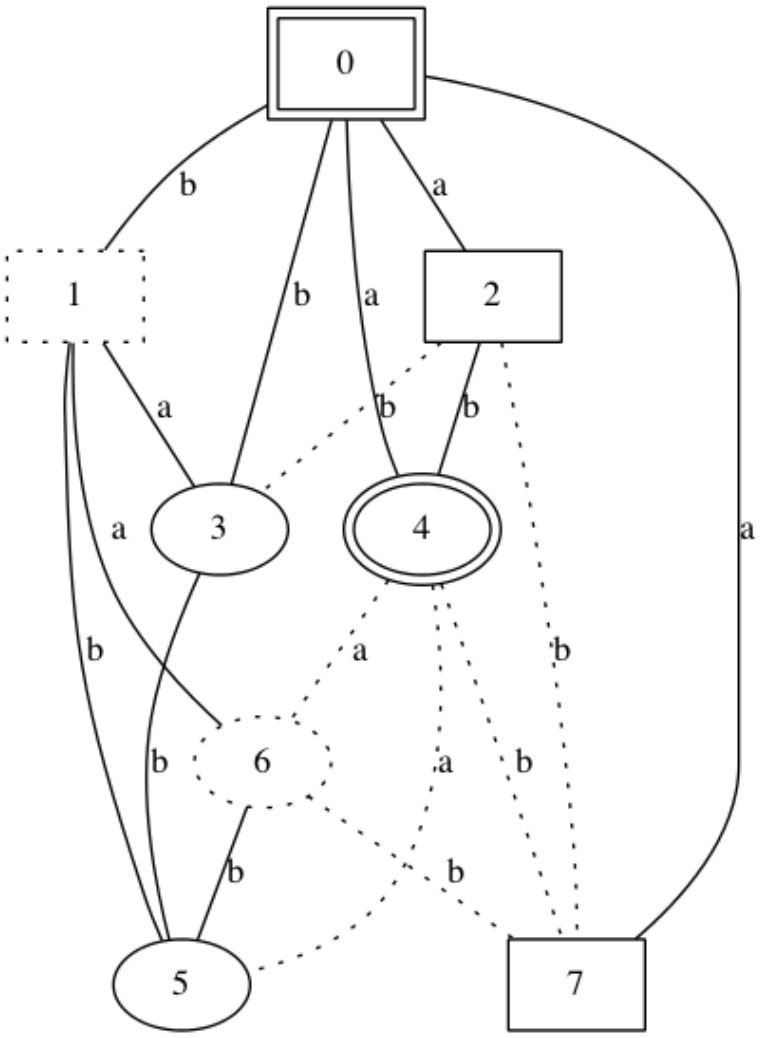

Fig. 5. Anonymised SN.

the private information of their users when the network is sold/shared/released. Anonymisation methods play a key role in this cases. The main purpose of these methods is to protect the sensitive (private) information of the users involved in the network while making the process of de-anonymisation more difficult. Current methods for anonymisation are not safe enough. We have formally introduced the `ADegree` constraint utilized in the process. The constraint seeks to protect sensitive information of the network avoiding issues of the current anonymisation processes. `ADegree` finds a network G' such that: a) private information is removed from G', making the network anonymous. As stated before, this process is not enough to protect the network from different attacks. We use anonymisation technique (i.e., adding fake edges), enforcing k-degree anonymous property over G'. b) G' has some modification with respect to the original network in order to make the reverse process of de-anonymisation harder. Enforcing this property implies that G' losses some interesting properties that were contained in the original network. The process of finding G' takes into account the utility of the network, that is, G' is as close as possible to original graph, where closeness is defined by users (a parameter of the constraint). We also show the implementation of the constraint using Gecode. The current implementation does not consider labelled graphs. Section 5.2 proposes an extension of the current implementation to handle labels whilst not introducing extra time complexity. We plan to extend the implementation to handle labels to compare our approach to the existing ones. We also plan to formally prove the following properties of `ADegree`.

Correct: it never removes values that are consistent with respect to its constraint.

Checking: it is singleton correctness (accept all satisfying assignments) and singleton completeness (reject all non-satisfying assignments).

Domain consistent: it removes all inconsistent values.

Finally, we plan to exploit all CP techniques in order to anonymise a social network under different type of attacks such as k-automorphism, l-neighbours and others. Also, we started investigating to provide a solution for preserving privacy under sequential releases of the same Social Network.

References

1. Lars Backstrom, Cynthia Dwork, and Jon Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 181–190, New York, NY, USA, 2007. ACM.
2. Smriti Bhagat, Graham Cormode, Balachander Krishnamurthy, and Divesh Srivastava. Class-based graph anonymization for social network data. *Proc. VLDB Endow.*, 2(1):766–777, August 2009.
3. Alina Campan and Traian Marius Truta. Privacy, security, and trust in kdd. chapter Data and Structural k-Anonymity in Social Networks, pages 33–54. Springer-Verlag, Berlin, Heidelberg, 2009.

4. P. Erdős and T. Gallai. Graphs with prescribed degrees of vertices (Hungarian). *Mat. Lapok*, 11:264–274, 1960.
5. Gecode Team. Gecode: Generic constraint development environment, 2006. Available from `http://www.gecode.org`.
6. Michael Hay, Gerome Miklau, David Jensen, Don Towsley, and Chao Li. Resisting structural re-identification in anonymized social networks. *The VLDB Journal*, 19(6):797–823, December 2010.
7. Andrei Lebedev, JooYoung Lee, Victor Rivera, and Manuel Mazzara. *Link Prediction using Top-k Shortest Distances*. Springer LNCS, 2017.
8. Joo Young Lee and Jae C Oh. Agent perspective social networks: Distributed second degree estimation. *Encyclopedia of Social Network Analysis and Mining*, pages 1–12, 2017.
9. JooYoung Lee. Reputation computation in social networks and its applications. 2014.
10. JooYoung Lee, Kontantin Lopatin, Rasheed Hussain, and Waqas Nawaz. Evolution of friendship: a case study of mobiclique. In *Proceedings of the Computing Frontiers Conference*, pages 267–270. ACM, 2017.
11. JooYoung Lee and Jae C Oh. A node-centric reputation computation algorithm on online social networks. In *Applications of Social Media and Social Network Analysis*, pages 1–22. Springer International Publishing, 2015.
12. Changchang Liu and Prateek Mittal. Linkmirage: How to anonymize links in dynamic social systems. *CoRR*, abs/1501.01361, 2015.
13. Kun Liu and Evimaria Terzi. Towards identity anonymization on graphs. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 93–106, New York, NY, USA, 2008. ACM.
14. Manuel Mazzara, Luca Biselli, Pier Paolo Greco, Nicola Dragoni, Antonio Marraffa, Nafees Qamar, and Simona de Nicola. *Social networks and collective intelligence: a return to the agora*. IGI Global, 2013.
15. Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. In *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy*, SP '09, pages 173–187, Washington, DC, USA, 2009. IEEE Computer Society.
16. Binh P. Nguyen, Hoa Ngo, Jihun Kim, and Jong Kim. Publishing graph data with subgraph differential privacy. In *Revised Selected Papers of the 16th International Workshop on Information Security Applications - Volume 9503*, WISA 2015, pages 134–145, New York, NY, USA, 2016. Springer-Verlag New York, Inc.
17. Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., New York, NY, USA, 2006.
18. Sergei Solonets. Adegree constraint implementation, 2017. Available at https://github.com/Solonets/ADegree.
19. Latanya Sweeney. K-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570, October 2002.
20. Bin Zhou and Jian Pei. Preserving privacy in social networks against neighborhood attacks. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, ICDE '08, pages 506–515, Washington, DC, USA, 2008. IEEE Computer Society.
21. Lei Zou, Lei Chen, and M. Tamer Özsu. K-automorphism: A general framework for privacy preserving network publication. *PVLDB*, 2(1):946–957, 2009.