2011 IEEE International Conference on Fuzzy Systems
June 27-30, 2011, Taipei, Taiwan

On Some Clustering Approaches For Graphs

Klara Stokes
Universitat Rovira i Virgili
Dept. of Computer Engineering and Maths,
UNESCO Chair in Data Privacy
Av. Països Catalans 26,
43007 Tarragona, Catalonia, Spain
Email: klara.stokes@urv.cat

Vicenç Torra
IIIA, Institut d'Investigació en Intel·ligència Artificial
CSIC, Consejo Superior de Investigaciones Científicas
Campus Universitat Autònoma de Barcelona,
08193 Bellaterra,
Catalonia, Spain
Email: vtorra@iiia.csic.es

***Abstract*—In this paper we discuss some tools for graph perturbation with applications to data privacy. We present and analyse two different approaches. One is based on matrix decomposition and the other on graph partitioning. We discuss these methods and show that they belong to two traditions in data protection: noise addition/microaggregation and k-anonymity.**

***Index Terms*—Data privacy, clustering, k-anonymity, microaggregation, graph**

I. Introduction

Graphs are useful mathematical structures for representing a large variety of data. In particular, they can be used to represent social networks and other kind of data where elements establish relationships with other elements.

In this paper we discuss methods for protection of data represented in terms of a graph. More specifically, we focus on problems in which the graph can be represented in an explicit way in terms of an adjacency matrix of manageable size.

The interest of this work is twofold. First, it is useful for protection of social networks with a limited number of nodes, as e.g. the data from a company or a university department. Naturally, this approach can be adapted to subsets of larger networks. Second, as these methods can be applied to a network that can be expressed in its entirety in a compact way, they permit us to analyse with some detail their effect as well as the effect of their parameterizations. In particular, we can study their effectiveness with respect to disclosure risk and information loss.

A. Our scenario in data privacy

We discuss in this section the scenario of data privacy we are considering in this paper. The scenario is basically the same as in previous research in the area of data privacy (see e.g. [1], [2]), although those papers focus on standard databases and not on relational data representable as graphs.

Scenario 1.1: A company or a national office wants to release data to a third party for doing some kind of analysis. Nevertheless, confidentiality should be ensured and so, no confidential information should be inferred by this third party from the data received.

In the framework of this scenario, existing tools for data privacy can be roughly classified in two broad classes:

- Cryptographic or computation oriented methods. They are methods based on cryptographic tools which try to avoid any disclosure at all. These methods are developed ad hoc for a particular analysis. That is, given a particular computation – a function of the data, a particular protocol is built which computes the function without giving to the data suppliers more information than the result of the function. As such, these methods are computation oriented, as they need to know and are based on the function to be computed.
- Perturbative or data oriented methods. These methods try to avoid disclosure manipulating the original data set and introducing some noise into the set. Then the perturbated data is released to the third party who can apply any analysis to the data. In this way, there is no need of knowing the function to be computed. That is why these methods are data oriented. Disclosure is (partially) avoided because of the perturbation inflicted to the data. Nevertheless, some disclosure risk is inherent in the data and, therefore, needs to be quantified. At the same time, the perturbation modifies the data and some information loss occurs. So, information loss also needs to be quantified.

Following [1], [3], we focus on the data oriented approach. That is, we focus on perturbative methods that add some kind of noise to the data. As stated above, the evaluation of these methods requires the quantification of information loss and disclosure risk. Then, we consider that a method has a good performance (i.e., it is a good data protection method) if for a particular type of data it achieves a good compromise between disclosure risk (i.e., the risk of disclosing sensitive information) and information loss (i.e., the amount of *data utility* that is lost when data is perturbated).

Among all possible data types, this paper focus on data protection tools for data representable in terms of graphs. The literature already presents some tools for data protection of graph structures. See e.g. the survey in [4]. There are tools [5] for graph perturbation based on random addition and deletion of nodes, and others [6] based on k-anonymity.

In this paper we focus on tools for small graphs. More specifically, in tools for graphs that can be representable in their entirety by a adjacency matrix or an equivalent structure and small enough to be stored and operated in memory. That is, we focus on tools for graph perturbation to be applied to

978-1-4244-7317-5/11/$26.00 ©2011 IEEE

graphs of limited size. Then, we develop methods based on the decomposition of the matrix and on similarities between the nodes of the graph. This diverges from previous works.

The goal of this paper is to present a set of tools for data privacy that will be further studied in detail for analysing its effectiveness. We discuss the appropriateness of our approach and illustrate it with some examples. Then, in a further research, we can develop approximate solutions for larger datasets, or apply them to subsets of larger sets.

The structure of the paper is as follows. In Section II we review some basic concepts that are needed in the rest of the paper. Then, in Section III we present some approaches for graph perturbation. The paper finishes with some conclusions.

II. Preliminaries

In this section we review some concepts that are needed later on in this paper. We focus on some tools we use later for operating with graphs. In particular, in methods of matrix decomposition.

A. Matrix decomposition

We will denote by U^H the conjugate transpose of U. That is, U^H is the transpose of the conjugate matrix. Recall that the conjugate of the complex number $a + bi$ is $a - bi$. When U only contains real numbers, U^H is the standard transpose of U.

A matrix M is said to be unitary when $MM^T = I_n$. Here I_n stands for the diagonal matrix of dimension n.

The singular value decomposition of a matrix M consists of a factorization of M in three matrices U, D and V as follows:

$$M = UDV^H$$

where U and V are unitary matrices and where D is a nonnegative diagonal matrix.

Here, if M is an $m \times n$ matrix, then U is an $m \times m$ matrix, D is an $m \times n$ matrix, and V is an $n \times n$ matrix.

The eigendecomposition or spectral decomposition of a square matrix M of dimension n consists of the factorization of M in terms of two matrices V and D as follows:

$$M = VDV^{-1}$$

where V is an $n \times n$ matrix with the eigenvalues of M, and D is an $n \times n$ diagonal matrix with the eigenvalues of M on the diagonal.

Both decompositions are related for some particular matrices. In particular, if M is symmetric and positive definite, then the spectral decomposition corresponds to a singular value decomposition.

III. Graph perturbation

In this section we discuss two different approaches for graph perturbation. One is based on matrix decomposition, and the other on graph partitioning. Both follow the tradition of perturbation methods. Nevertheless, they can be seen as belonging to two different *traditions*. We will discuss this later in Section III-C.

A. Methods based on matrix decomposition

In statistics a dimension reduction is a reduction of the number of random variables under consideration. The dimension reduction can be performed using the spectral decomposition or the singular value decomposition (SVD) of the matrix representing the data. If we apply the same methods to the adjacency matrix of a graph, then we accomplish a perturbation of the matrix. Appropriate selection of the eigenvalues in the spectral method and of the singular values in the SVD method will reduce the *complexity* of the matrix so that the most significant features of the structure is kept. In this way it is possible to perturbate the graph while keeping its most significative edges.

The rationale of the approach is to perform a transformation similar to random addition and deletion of edges (as in e.g. [5]) but in a more learned way. We define the approach below. We start with the method based on the spectral decomposition.

Definition 3.1: Let $G = (V, E)$ be a graph and let M be its adjacency matrix, let p be a parameter such that $0 \leq p \leq |V|$. Then, the SD-graph protection method consists in (i) computing the spectral decomposition of M: let $M = UDU^{-1}$ be the spectral decomposition of M; (ii) returning $M' = UD'U^{-1}$ where D' includes only the first p eigenvalues of D.

Here, as usual in spectral decomposition, we assume that the eigenvalues of D are arranged in descending order with respect to the size of their absolute value.

We now define the method based on singular value decomposition.

Definition 3.2: Let $G = (V, E)$ be a graph and let M be its adjacency matrix, let p be a parameter such that $0 \leq p \leq |V|$. Then, the SVD-graph protection method consists in (i) computing the singular value decomposition of M: let $M = U\Sigma W^H$ be the singular value decomposition of M; (ii) returning $M' = U\Sigma' W^H$ where Σ' includes only the first p values of the diagonal of Σ, where these are arranged in descending order.

As we saw in the introduction, the SVD and the spectral decomposition are related. For undirected graphs we get the following proposition.

Proposition 3.3: For undirected graphs the methods defined in Definition 3.1 and Definition 3.2 are equivalent.

Proof: The adjacency matrix of an undirected graph is symmetric and real, but in general not positive definite. The singular values of the matrix are the absolute values of its eigenvalues. This implies that for undirected graphs the two methods are equivalent. ■

For both methods, the smaller the number p, the larger the distortion inflicted to the graph. Naturally, when p equals the dimension of the matrix (i.e., $p = |V|$), data protection methods cause no distortion to the graph; in contrast, when $p = 0$, the distortion is maximal as the graph is completely distorted. Note that for $p = 0$, all edges are lost.

Therefore, the parameter p controls the degree of noise. This is in line of most perturbative data protection methods for databases. Note that e.g. the parameter k in microaggregation [7] permits the user a rank of results from maximal

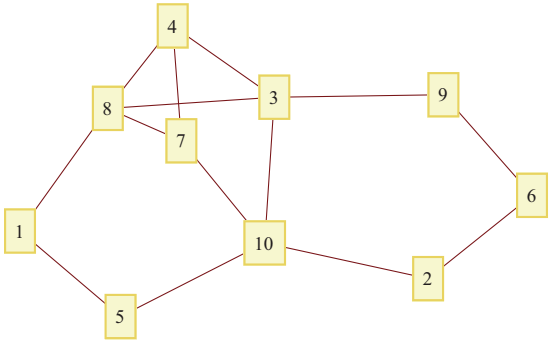
distortion (all records are replaced by the same value) to no distortion at all (all records are published as they are). Similar situation occurs with the different variants of rank swapping [8].

Although both methods are a natural way to implement a perturbation that introduces noise to the original data, they have two drawbacks.

The first one is that while the original matrix M contains only binary values to represent presence or absence of an edge, the matrix obtained as output from the method has coefficients with values in the real numbers. They are not restricted to [0,1] and can even be negative. This requires to develop a post processing in which the resulting matrix is transformed into a proper graph. For example, this binarization process could consist in transforming M' into a set of probabilities, and then using these probabilities for building a new matrix $\tilde{M}'$. Another approach for binarization consists in assigning zero or one according to a threshold. This latter approach has been applied in Example 3.4.

The second one is that, once the binarization process is applied, the *graph* represented by M' is quite similar to M for several of the parameterizations of the methods. That is, although M' is no longer a binary matrix, zero values in M result into values near zero in M', and unitary values in M result into values near one in M'. For example, for the matrix in Example 3.4, using a binarization with a threshold of 0.5, we have that we have to perform a dimension reduction of at least 6 ($p = 4$) in order to obtain a graph which is different from the original one. With $p = 4$ we remove two edges, with $p = 3$ we remove 8 edges and with $p = 2$ we remove 16 edges. With $p = 1$, 23 edges are changed (7 added and 16 removed) and, of course, all 28 edges are removed taking $p = 0$.

Example 3.4: This is a graph on 10 vertices with 14 edges:



This is the adjacency matrix of the graph:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

The singular values of this matrix are approximately

$$\{3.16, 2.52, 1.83, 1.63, 1.56, 1.28, 0.83, 0.77, 0.41, 0.31\}.$$

We applied first dimension reduction using SVD on the adjacency matrix of the graph. All values of p between 0 and 10 were considered. Below we list the correlations between the original matrix and the result as a function of p, using the function implemented in R [9].

$dimensionality\ reduction(1-p)$	$correlation$
0	1
1	0.4283313
2	0.475191
3	0.3822435
4	0.4283313
5	0.3360108
6	0.3822435
7	0.28843
8	0.2375955

Then we applied a binarization process to the result with a threshold of 0.5. Below we list the correlations between the original matrix and the binarized matrix as a function of p. For $p = 10$ no correlation can be computed because the perturbated graph has no edge.

$dimensionality\ reduction(1-p)$	$correlation$
0	1
1	1
2	1
3	1
4	1
5	1
6	0.948448
7	0.7953949
8	0.4516129
9	0.5663766
10	$--$

The selection of the threshold was done heuristically, defined as the mid point between 0 and 1 with respect to the perturbation of the matrix. This selection results into a rather conservative selection. For $p = 2$, except a threshold of 0.4 that leads to 13 changes in the matrix, all other thresholds lead to the same or more changes (with maximums of 49 and 20 changes for thresholds of 0.1 and 0.9).

When constructing a method for data perturbation with the ambition to obtain data privacy it is important to verify that the perturbation is hard to reverse, since otherwise the method implies no privacy, or at most limited privacy. The following arguments are made under the assumption that the graphs are undirected and simple (i.e. without multiedges and loops). Research on spectral graph theory has shown that 'almost all' graphs are uniquely determined by their spectra [10], so that in many cases given a sequence of eigenvalues, there is only one graph which has an adjacency matrix with these

eigenvalues. This is so due to the difficulties to find a set of eigenvectors which brings the diagonal matrix to a space where the matrix is symmetric, with entries either 0 or 1 and with the diagonal all 0. Suppose that we apply to a graph one of the methods for graph perturbation defined above. Before applying the binarization process, the result is a matrix M with p of the eigenvalues equal to the largest p eigenvalues of the original adjacency matrix A, ordered by their absolute value. The rest of the eigenvalues of M are all 0. The eigenvectors of M are the same as the eigenvectors of A. Counting the number of the eigenvectors of M which are 0, we can bound the number of removed eigenvectors. The absolute value of a removed eigenvalue (i.e. any eigenvalues of A which is not an eigenvalue of M) is smaller than the smallest absolute value of all the eigenvalues of M. Given M this significantly restrict the possible choices of A. If we also remember that since the diagonal of an adjacency matrix of a simple undirected graph is always 0, so is its trace, then it is clear that in some cases it is very easy to given M deduce A. For example when only one eigenvalue is removed, the missing eigenvalue is minus the sum of the other eigenvalues.

This observation highlights the importance of binarization process. Indeed, through the binarization process perturbations are introduced into the *eigenvectors* of the graph. This destroys the possibility to easily reverse the perturbation process. Important aspects of the privacy therefore relies on the binarization process, implying that the choice of binarization process could be most significant for the degree of privacy supplied by the method. Since the information loss of course also depends on the binarization process, the choice of the binarization process should be further investigated in the future.

B. Methods based on graph partitioning

The literature for database protection includes a large number of methods for achieving k-anonymity [2], [11], [12]. A file is said to be compliant with k-anonymity if for each record there are other $k-1$ indistinguishable records.

k-anonymity is used for data protection based on the following rationale:

> No disclosure takes place when any individual is indistinguishable of $k-1$ other different individuals.

Then, methods transform the original data file so that only sets of k-indistinguishable records are found in the data. Naturally, all such methods use k as one of the parameters. These methods act by means of clustering, generalizations and suppressions.

In the framework of k-anonymity, disclosure risk is usually not analysed but taken for granted once k is settled. This is so because the parameter k is assumed to be a threshold of the satisfactory disclosure risk. So, only information loss is evaluated here.

Methods for graph partitioning can be used for achieving protection following the idea of k-anonymity. In its simplest form, graph partitioning divides the vertices in two sets of equal size such that the number of edges between these two parts is minimized. In a more general formulation, the vertices are divided in p disjoint subsets of equal size.

Therefore, we can conceive a kind of k-anonymity from the results of graph partitioning. Let $G = (V, E)$ represent a graph, and let $\Pi_V = \{V_1, \ldots, V_p\}$ represent a partition of the edges of this graph into p clusters. Then, we consider the vertices of each cluster V_i as indistinguishable. So, given p, this results in $k = |V|/p$-anonymity. At this point, two alternatives can be foreseen.

1) Direct partitioning: In this case, we apply a partitioning algorithm to the original graph (i.e., to the original matrix). This causes clusters to be built on the basis of good connectivity. That is, the method detects sets of vertices that are well connected among themselves but with a low connectivity with the other ones.

This approach has an inconvenience. The partitioning algorithm does not distinguish between the different roles that vertices might have in well connected regions. To illustrate this point, let us consider the following example.

Let $G = (V, E)$ be a graph representing the relationships among members of a university. Let the vertices V represent the staff of the university and edges E represent mails and other types of communication among staff members. In this case, for an appropriate value of p, a graph partitioning algorithm will form p clusters. Such clusters shall roughly correspond to departments. Note that communication links between department members is likely to be more intense than communication links between members belonging to different departments.

So, in general, this approach is unable to detect and distinguish that within the department, some of the members might be administrative staff, other senior research staff, and other students. The approach proposed in the next section focuses on the detection of such roles.

2) Indirect partitioning: In this case, the partitioning algorithm is applied to a similarity matrix computed from the graph. Therefore, this is a two step process.

- First, we define a matrix $S : V \times V \to [0, 1]$ representing the edge similarity between the pairs of vertices in the graph with $|V| = n$.

  $$S(v_i, v_j) = \text{similarity}(v_i, v_j)$$

  where the similarity function should be selected in order to give a satisfactory clusterization.
- Second, we apply a graph partitioning algorithm to the matrix S.

While in the direct approach the algorithm clusters together well connected vertices, in the indirect approach we cluster vertices with a similar role in the graph. In some sense, this is similar to the approach in [6] which discusses the isomorphism between edges in the graph.

We have considered two distinct definitions of similarity between vertices: one based on the l_1 or Manhattan distance and the other based on which neighbors the vertices have in common. We define these two similarities below and discuss their properties. We represent the edges of a vertex v_i by the

row vector $A(i)$ of the adjacency matrix A (or equivalently, the column vector $A(j)$) corresponding to the vector.

Definition 3.5: Using the Manhattan or the l_1 distance, the similarity between two vertices v_i and v_j with edges $A(i)$ and $A(j)$ in a graph on n vertices is defined as

$$\text{similarity}_{l_1}(v_i, v_j) = n - \sum_{k=1}^{n} |A(i,k) - A(j,k)|$$

$$= \sum_{k=1}^{n} (1 - |A(i,k) - A(j,k)|) .$$

Definition 3.6: The 2-path similarity between two vertices v_i and v_j with edges $A(i)$ and $A(j)$ in a graph on n vertices is defined as

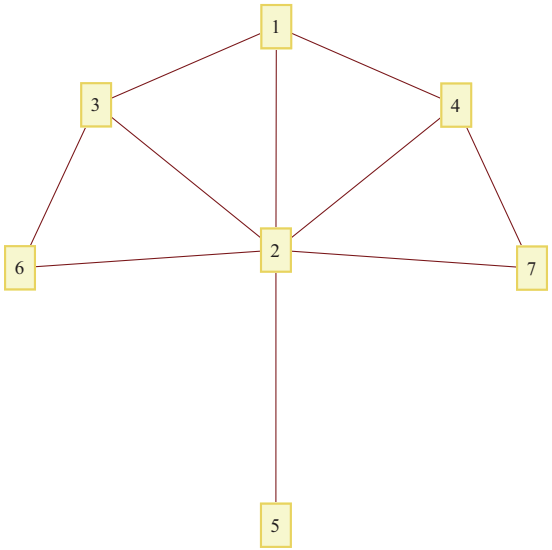
$$\text{similarity}_{2-path}(v_i, v_j) = \sum_{k=1}^{n} A(i,k)A(j,k).$$

Observe that since an adjacency matrix of an undirected graph is symmetric, the ith row is the ith column transposed.

The 2-path similarity is augmented with one for every neighbor the two vertices have in common. The similarity based on the Manhattan distance is also augmented with one for every neighbor the two vertices have in common, but it is augmented with one for every non-neighbor they have in common as well.

The 2-path similarity highlights the fact that similarity between vertices is based on the existing edges, and not on the missing edges. The following example illustrates this point.

Example 3.7: This is a graph with 7 vertices and 10 edges:



This is the adjacency matrix of the graph:

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

The distance from vertex 1 to the other vertices is listed below.

vertex	similarity$_{l_1}$	similarity$_{2-path}$
2	2	2
3	3	1
4	3	1
5	5	1
6	6	2
7	6	2

The example illustrates the differences between the two similarities. It can be seen that while the similarity between vertex 1 and vertex 5 is much larger for the Manhattan distance than the similarity between vertex 1 and 2, in the 2-step similarity we have the reversal.

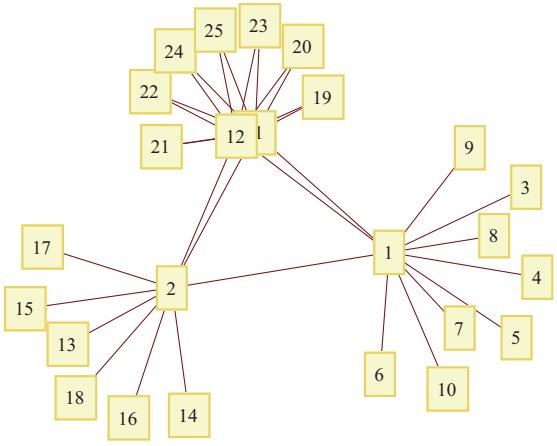
This is due to the fact that vertices 1 and 5 share four non-neighbours, while 1 and 2 do not share any of them. In contrast, vertices 1 and 2 share two neighbors and vertices 1 and 5 share only one. The 2-step similarity focuses on the common neighbors.

In relation to the interest of the approach of indirect partitioning with respect to the one of direct partitioning, let us consider the following scenario.

Example 3.8: Let us consider a social network where vertices can either represent individuals or mass media. While in the direct approach, individuals and media might be clustered together according to communities of news producers and consumers, in the indirect approach, producers and consumers can be kept disaggregated. That is, consumers with similar interests (reading newspaper n_1, n_2 and listening radios r_1, r_2) will be aggregated together in the same cluster. At the same time, media with a similar audience will also be clustered together (n_1, n_2, r_1 and r_2 will form a cluster).

We include below an example that illustrates this situation with a graph, and showing the results of the approach for indirect partitioning.

Example 3.9: This is a graph on 25 vertices with 33 edges:



In this example vertices 1, 2, 11, and 12 represent the mass media and the rest of the vertices represent individuals. Edges from individuals to mass media represent that the individuals access/use the mass media while edges between pairs of mass media represent interactions among them. Graph partitioning

using the similarity based on the Manhattan distance results in the following partition:

$$\{1, 2, 3, 4, 5, 6, 11, 12\}$$

$$\{8, 9, 10, 13, 14, 15, 16, 17, 18\}$$

$$\{7, 19, 20, 21, 22, 23, 24, 25\}$$

and graph partitioning using the 2-path similarity results in the following partition:

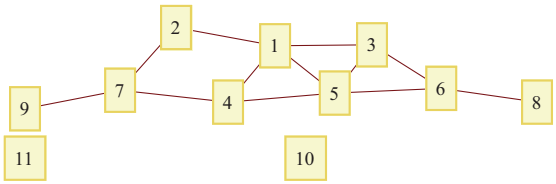
$$\{1, 2, 19, 20, 21, 22, 23, 24, 25\}$$

$$\{11, 12, 13, 14, 15, 16, 17, 18\}$$

$$\{3, 4, 5, 6, 7, 8, 9, 10\}$$

It can be seen that this partition includes in the partition elements the consumers of a (group) of media while the media themselves are disaggregated from the individuals. In contrast, graph partitioning using Manhattan distance does not show this behaviour as the media are clustered with their individuals. For example, the partition of the Manhattan distance contains 1, 2, 3, 4, 5, 6, 11, 12 but the 2-path distance contains 3, 4, 5, 6, 7, 8, 9, 10. Note that this latter cluster contains all vertices related to 2, but not the vertice 2. In contrast, the former contains a mix of nodes connected to 2, with node 2.

Example 3.10: In the graph below there are two leafs, the vertices 8 and 9. We want the vertices of the graph to be clusterized so that vertices with a similar role in the graph end up in the same cluster. What we mean with this depends on what we mean when we say that two vertices play a similar role. In this case, if the role is to be a leaf, then we probably want 8 and 9 to go together. However, if the role is to be connected to more or less the same vertices, then vertices 8 and 9 should be in different clusters.



Clustering the graph with respect to the Manhattan distance using the partition algorithm in METIS we obtain the partition

$$\{1, 5, 7\} \quad \{2, 8, 9, 10, 11\} \quad \{3, 4, 6\}$$

and if the clusterization is performed with respect to the 2-path distance, then the same partition algorithm gives us

$$\{8, 10, 11\} \quad \{2, 3, 4, 9\} \quad \{1, 5, 6, 7\}.$$

If we use the partition algorithm from Mathematica instead, in the last case we obtain

$$\{1, 7, 5, 6, 3, 8\} \quad \{2, 4, 9\} \quad \{10\} \quad \{11\}.$$

This illustrates that if the partition algorithm is calibrated in an adequate way (in this case it means that the number of partitions should be adequate), then the 2-path similarity creates clusters which collect vertices which more or less are connected to the same vertices, while the similarity based on the Manhattan distance rather tends to cluster vertexes together which are similar also in their non-existing edges, like for example two leafs or two isolated vertices like vertices 10 and 11. Indeed it is clear that the similarity based on the Manhattan distance considers similarity with respect to non-existing edges equally important as similarity with respect to existing edges.

Graph partitioning does not fully define a method compliant with k-anonymity. The correct application of this approach to a graph requires that the resulting partition satisfies two requirements: (i) that the number of vertices in a cluster is large enough; (ii) that the edges between vertices are reconstructed.

For solving the first issue, we propose the use of graph partitioning algorithms that lead to solutions with a similar number of vertices in all partitions. Then, if we require at least k vertices in each partition element, we should select a number of clusters less than $|V|/k$. The algorithm for graph partitioning implemented in METIS [13], [14] proceeds in this way.

As forcing solutions of this form might result into unnatural clusters for large graphs, an alternative approach is to proceed in a two step process. First, partitioning the vertices without any constraint (or a loose constraint) on the number of vertices in each cluster (the clustering algorithm in Mathematica proceeds in this way), and, then, refine the largest clusters applying a more constrained clustering algorithm.

The second element to be considered is about edge reconstruction. Once clusters are defined, the original edges have to be updated. First, vertices within the cluster are connected among themselves according to a pre-specified algorithm. Then, vertices should be connected to the other clusters, and all vertices from one cluster should be connected in the same way to vertices in another cluster. The decision of a connection can be done according to the majority of presence or absence of edges between the corresponding vertices.

The following proposition is crucial for how to choose the connections for the reconstruction between vertices in the same cluster.

Proposition 3.11: We call a cycle of length 3 a triangle. If the original graph is triangle-free, then the 2-path distance will cluster the graph so that the vertices in the same cluster are not connected.

Proof: We may suppose that two vertices a and b in the same cluster have at least one neighbor c in common, making a and b similar. Suppose that there is an edge between a and b. Then, since we have the path bca, there is the cycle $abca$ of length 3, that is, a triangle. This implies that in triangle-free graphs there are no edges between the vertices in the same cluster if the cluster was made using 2-path similarity. ■

The proposition implies that whenever the graph is triangle-free, assuming that it is desirable to preserve this property, the reconstruction of the graph from the clusters should be such that vertices in the same cluster are not connected.

Observe that if we cluster with respect to the similarity based on the Manhattan distance, Proposition 3.11 does not

apply, since two vertices in the same cluster not necessarily have any neighbors in common. Indeed, the similarity based on the Manhattan distance groups all isolated vertices together.

C. Discussion on graph perturbation

We have introduced two different families of methods for graph perturbation. One family consists on methods based on matrix decomposition, and the second one consists on methods based on graph partitioning.

While both methods perturbate the original graph resulting into a new graph with changes on the edges, they can be seen as belonging to two different traditions of perturbative approaches.

Methods based on matrix decomposition, with the parameter p, result into methods which try to find a good trade-off between information loss and disclosure risk. That is, on the tradition of noise addition and microaggregation. In contrast, methods based on graph partitioning follow the tradition of k-anonymity.

IV. Concluding remarks

In this paper we have presented some perturbation methods for graphs of limited size. We have discussed briefly some of the properties of the approaches, and we have given some examples of their behaviour. Further research should concentrate on analysing the properties of these methods further, and it is necessary to perform additional experiments for graphs of larger sizes. Future research should also include a comparative study of the results obtained by applying the different methods.

Acknowledgments

The first author was partially supported by the Spanish Government through projects TIN2009-11689 "RIPUP", CONSOLIDER INGENIO 2010 CSD2007-00004 "ARES", and FPU grant (BOEs 17/11/2009 and 11/10/2010) and by the Government of Catalonia under grant 2009 SGR 1135.

The second author was partially supported by the Spanish MEC (projects ARES – CONSOLIDER INGENIO 2010 CSD2007-00004 – and eAEGIS – TSI2007-65406-C03-02). The authors are with the UNESCO Chair in Data Privacy, but their views do not necessarily reflect those of UNESCO nor commit that organization.

References

[1] J. Domingo-Ferrer and V. Torra, "A quantitative comparison of disclosure control methods for microdata," in Confi dentiality, Disclosure and Data Access: Theory and Practical Applications for Statistical Agencies, P. Doyle, J. I. Lane, J. J. M. Theeuwes and L. Zayatz, Eds. North-Holland, 2001, pp. 111-134.

[2] L. Sweeney, "k-anonymity: a model for protecting privacy," Int. J. of Unc., Fuzz. and Knowledge Based Systems, vol. 10:5, pp. 557-570, 2002.

[3] W.E. Yancey, W.E. Winkler and R.H. Creecy, "Disclosure risk assessment in perturbative microdata protection," in Inference Control in Statistical Databases, J. Domingo-Ferrer, Eds. Lecture Notes in Computer Science 2316, 2002, pp. 135-152.

[4] D. Nettleton, V. Torra, "Data privacy for simply anonymized social network logs represented as graphs," unpublished.

[5] M. Hay, G. Miklau, D. Jensen, P. Weis, S. Srivastava, "Anonymizing social networks," Technical Report 07-19, University of Massachusetts Amherst, 2007. Published at VLDB 2008.

[6] K. Liu, E. Terzi, "Towards Identity Anonymization of Graphs," In proceedings of SIGMOD, 2008.

[7] J. Domingo-Ferrer, J.M. Mateo-Sanz, "Practical data-oriented microaggregation for statistical disclosure control," IEEE Trans. on Knowledge and Data Engineering, vol. 14:1, pp. 189-201, 2002.

[8] J. Nin, J. Herranz, V. Torra, "Rethinking Rank Swapping to Decrease Disclosure Risk," Data and Knowledge Engineering, vol. 64:1 pp. 346-364, 2008.

[9] T. Butts, Package 'sna', Project R, 2011.

[10] E.R. van Dam, W.H. Haemers, "Spectral characterizations of some distance-regular graphs," Journal of Algebraic Combinatorics, vol. 15:2 pp. 189–202, 2002.

[11] P. Samarati, "Protecting Respondents' Identities in Microdata Release," IEEE Trans. on Knowledge and Data Engineering, vol. 13:6, pp. 1010-1027, 2001.

[12] P. Samarati, L. Sweeney, "Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression," SRI Intl. Tech. Rep, 1998.

[13] http://glaros.dtc.umn.edu/gkhome/metis/metis/overview

[14] G. Karypis, V. Kumar, "Multilevel k-way Partitioning Scheme for Irregular Graphs," J. Parallel Distrib. Comput. vol. 48:1,pp. 96-129, 1998.