Using Genetic Algorithms for Attribute Grouping in Multivariate Microaggregation

Jordi Balasch-Masoliver[1], Victor Muntés-Mulero[2] and Jordi Nin[1]*
[1]Department of Computer Architecture,
Technical University of Catalonia - BarcelonaTECH (UPC),
Campus Nord UPC, C/Jordi Girona 1-3
08034 Barcelona, (Catalonia, Spain)
{jbalasch, nin}@ac.upc.edu
[2]CA Technologies,
Campus Nord UPC, C/Jordi Girona 1-3
08034 Barcelona, (Catalonia, Spain)
Victor.Muntes@ca.com

April 3, 2013

Abstract

Anonymization techniques that provide k-anonymity suffer from loss of quality when the data dimensionality is high. Microaggregation techniques are not an exception. Given a set of records, attributes are grouped into non-intersecting subsets and microaggregated independently. While this improves quality by reducing the loss of information, it usually leads to the loss of the k-anonymity property, increasing entity disclosure risk. In spite of this, grouping attributes is still a common practice for data sets containing a large number of records. Depending on the attributes chosen and their correlation, the amount of information loss and disclosure risk vary. However, there have not been serious attempts to propose a way to find the best way of grouping attribute. In this paper, we present GOMM, the Genetic Optimizer for Multivariate Microaggregation which, as far as we know, represents the first proposal using evolutionary algorithms for this problem. The goal of GOMM is finding the optimal, or near-optimal, attribute grouping taking into account both information loss and disclosure risk. We propose a way to map attribute subsets into a chromosome and a set of new mutation operations for this context. Also, we provide a comprehensive analysis of the operations proposed and we show that, after using our evolutionary approach for different real data sets, we obtain

*Corresponding author contact information: nin@ac.upc.edu, Phone:+34 93 401 6995 and Fax: +34 93 401 7055

better quality in the anonymized data comparing it to previously used adhoc attribute grouping techniques. Additionally, we provide an improved version of GOMM called D-GOMM where operations are dynamically executed during the optimization process to reduce the GOMM execution time.

Keywords: Genetic Clustering algorithms Multivariate Microaggregation Attribute Selection

1 Introduction

The necessity for keeping data set available for data mining while preserving some reasonable degree of privacy has given rise to statistical disclosure control techniques [10, 12, 27]. A common way of ensuring privacy without losing information is by ensuring that the released data is *k-anonymous* [23, 24], that is, that for any record in the set, there are at least $k - 1$ records that are indistinguishable from it. Microaggregation [3, 22] is one of the most common methods used to obtain k-anonymity for numerical data: groups of k nearest records, which are mapped as points in a multidimensional space defined by the attribute columns, are identified and substituted by their centroid.

The increase of information stored by most enterprises and organizations nowadays usually entails the increase of the complexity of data schemas [15]. Among other aspects, this typically involves an increase on the number of attributes per record, making the number of dimensions used to map the elements in the data set larger. The more dimensions the microaggregation problem deals with, the further the points mapped in this multidimensional space. Thus, microaggregating points that are too far implies that these are substituted by centroids which are too distant, which usually implies a significant loss of information [1].

In order to diminish the effect of high dimensionality, data sets are usually partitioned into disjoint attribute sets, so that each partition only preserves the values of certain attributes for each record. Afterwards, microaggregation is used in each partition separately. This process is known as *multivariate microaggregation*. While this reduces the information loss, it prevents from guaranteeing k-anonymity: two records that are clustered together using their values on a subset of their attributes might not be clustered together if another set of attributes are used as a criterium [17]. With this, we cannot ensure that these two records will not be undistinguishable anymore. Nevertheless, partitioning techniques are commonly used in order to preserve data utility.

As studied in [18], when protecting a data set using multivariate microaggregation, the way in which the data is split to form groups is highly relevant with regard to the degree of privacy achieved. That is, assuming a data set containing subsets of correlated attributes, if we group correlated attributes together and groups are, therefore, non-correlated, applying microaggregation to each partition will probably preserve a high information utility, while it will probably imply a complete loss of the k-anonymity property. On the contrary, grouping non-correlated attributes, may preserve k-anonymity better, at the cost of losing information utility. Achieving the best trade-off between information loss and entity disclosure risk by choosing the right attribute partitioning is still an open problem.

In this paper, we present a novel approach in order to decide the optimal, or near-optimal, attribute partitioning. Our technique is based on the use of genetic algorithms in order to explore the vast space of all possible attribute groupings. We provide a set of new mutation operations that, based on the mapping of a possible attribute grouping into a chromosome, allow exploring the specific search space presented in this scenario. We present a comprehensive analysis of our operations and show that they can overcome previously used ad-hoc attribute partitioning techniques, using well-known data sets.

The remainder of the paper is organized as follows: Section 2 we introduce the microaggregation anonymization scenario and the basic. Later, in Section3 we describe GOMM, our novel proposal for attribute grouping using genetic algorithms. Afterwards, Section 4 presents a complete analysis of GOMM operations. Section 5 performs a large number of experiments with real datasets to analyze GOMM performance. Finally, the paper finishes with some conclusions and future work.

2 Anonymization Preliminaries

In this section, we describe the typical anonymization scenario as well as some basic concepts about microaggregation.

2.1 Problem Statement

A data set R is a collection of records where each record r represents a point in a multidimensional space defined by the number of attributes. The attributes a_i can be classified in three non-disjoint categories: *identifiers* which unambiguously identify the individual (*e.g.* the passport number), *quasi-identifiers* which can identify the individual when some of those attributes are combined (*e.g.* age or postal code) and *confidential* attributes which contain sensitive information about the individual (*e.g.* salary).

When considering this classification, a data set R is defined as $R = id||a_{nc}||a_c$, where id are the identifiers, a_{nc} are the non-confidential quasi-identifier attributes, and a_c are the confidential attributes. Normally, before releasing a data set R with confidential attributes, a protection method ρ is applied, leading to a protected data set R'. Indeed, we assume the following typical scenario depicted in Figure 1: (i) identifier attributes in R are either removed or encrypted, therefore $R' = a'_{nc}||a'_c$; (ii) confidential attributes a_c are not modified, and so we have $a'_c = a_c$; (iii) the protection method itself is applied to non-confidential quasi-identifier attributes, in order to preserve the privacy of the individuals whose confidential data is being released. Therefore, we have

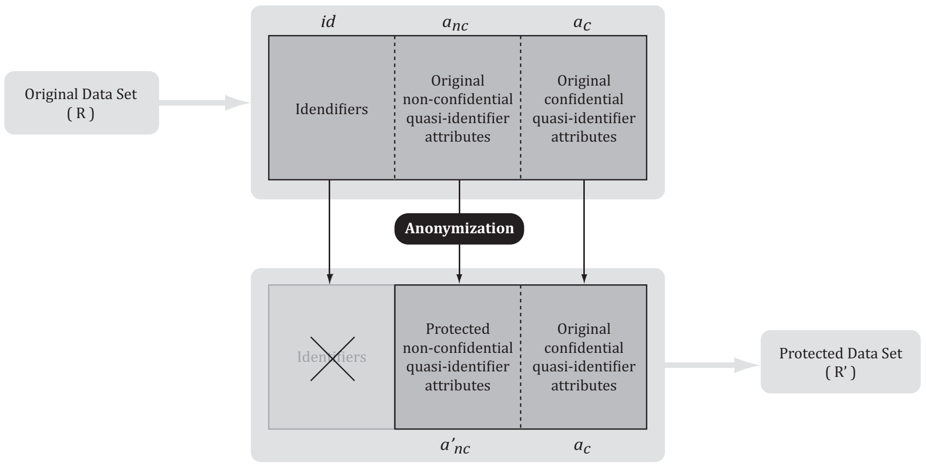

id
a_{nc}
a_c
Original Data Set (R)
Idendifiers
Original non-confidential quasi-identifier attributes
Original confidential quasi-identifier attributes
Anonymization
Idendifiers
Protected non-confidential quasi-identifier attributes
Original confidential quasi-identifier attributes
Protected Data Set (R′)
a'_{nc}
a_c

Figure 1: Graphical description of the data set protection and release process.

$a'_{nc} = \rho(a_{nc})$. This scenario allows third parties to have precise information on confidential data without revealing to whom the confidential data belongs.

In this scenario, as shown in Figure 2, an intruder might try to re-identify individuals by obtaining the non-confidential quasi-identifier data (a_{nc}) together with identifiers (id) from other data sources. By applying record linkage between the protected attributes (a'_{nc}) and the same attributes obtained from other data sources (a_{nc}), the intruder might be able to re-identify a percentage of the protected individuals together with their confidential data (a_c). This is what protection methods try to prevent.

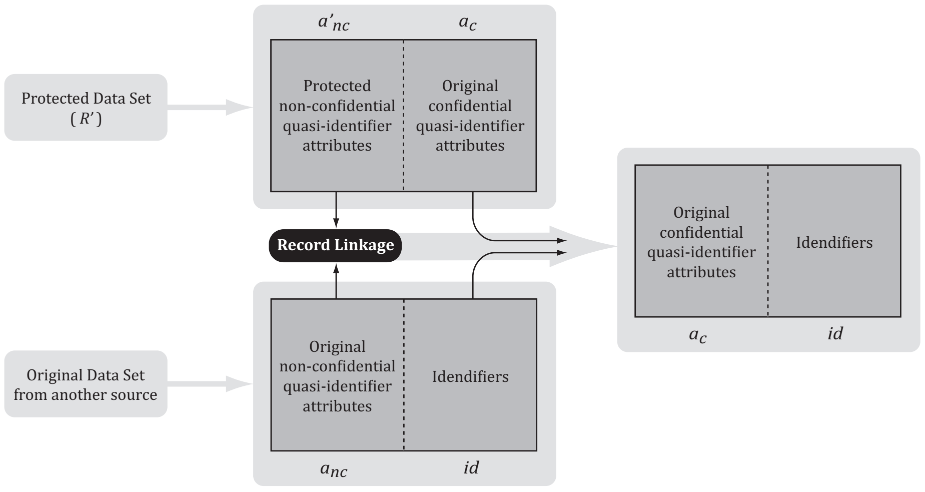
2.2 Microaggregation

The goal of any microaggregation method is to minimize the total Sum of the Square Error

$$SSE = \sum_{i=1}^{n} \sum_{r_j \in n_i} (r_j - \bar{r}_i)^T (r_j - \bar{r}_i), \qquad (1)$$

where r_j are the records of the data set R, n is the total number of clusters, n_i is the i-th cluster and $\bar{r}_i$ is the centroid of n_i. The restriction is $|n_i| \geq k$, for all $i = 1, \ldots, n$.

As we have explained before, when the number of attributes per record is large, *multivariate microaggregation* is used for reducing the information loss at the cost of increasing the disclosure risk. However, multivariate microaggregation has two main drawbacks. On the hand one, finding the optimal microaggregation, *i.e.* the optimal clusters configuration, is a NP-hard problem [19]. This problem has been widely studied, and a large variety of heuristic algo-



a'_{nc}
a_c
Protected Data Set (R')
Protected non-confidential quasi-identifier attributes
Original confidential quasi-identifier attributes
Record Linkage
Original confidential quasi-identifier attributes
Idendifiers
a_c
id
Original Data Set from another source
Original non-confidential quasi-identifier attributes
Idendifiers
a_{nc}
id

Figure 2: Intruder disclosure risk scenario.

rithms exist. As this problem is out of the scope of this paper, in this work we use MDAV (described later on), one of the most well-known and flexible multivariate microaggregation algorithms. On the other hand, grouping attributes properly is a problem as difficult as microaggregation itself. Since the search space grows exponentially with respect to the number of attributes, finding the most appropriate attribute grouping configuration is far from trivial. Opposite to microaggregation, this problem has been disregarded in the literature assuming that attributes are grouped considering some background knowledge, such as attribute correlations when we are interested on minimizing the information loss, or considering that the intruder only has access to a subset of attributes, then such attributes are grouped together.

In [17, 18] two approaches were presented for attribute selection, both based on the correlations among attributes. The first approach is based on cluster in the same group attributes that are highly correlated, minimizing in this way the intra-cluster distance between the centroid and the records. This approach has a low information loss but a high disclosure risk as in the case of univariate microaggregation. The second approach is based on clustering highly correlated attributes in different groups. The goal of this second approach is to increase the resulting anonymity. The rationale of this approach is the following one: If two records r_i and r_j are in the same cluster for some blocks, this means that the first attribute values of these records are more or less close to each other, and the same for the second attribute of the block, etc. Then, when we consider another block, if the j-th attribute of this new block is (highly) correlated with the j-th attribute of the latter block, records r_i and r_j will probably be close to each other as well, with respect to the attributes in the second block. Therefore, with some non-negligible probability, r_i and r_j will fall in the same cluster, reducing in this way the disclosure risk and obtaining a similar level of privacy to that obtained by the basic microaggregation. Of course, the information loss of this latter approach is larger than the former one.

MDAV algorithm. The MDAV (Maximum Distance to Average Vector) algorithm [7] is an heuristic algorithm for clustering records in a data set R, so that each cluster is constrained to contain at least k records. It works as follows. Firstly, given n records in the data set to be protected, it computes the average record $\bar{r}$ from all the records. Afterwards, it looks for s, the furthest record to the average record $\bar{r}$, and forms a cluster around it (this cluster contains s together with the $k-1$ closest records to it). Then, a new cluster is formed around the most distant record to s in the same way. When both clusters are formed, all the records belonging to such clusters are removed. This process is repeated until all the records are assigned to a cluster. Note that, the last cluster is built with the last records to be protected, so it might contain between k and $2k-1$ records.

3 GOMM: Genetic Optimizer for Multivariate Microaggregation

In this section we present GOMM, the Genetic Optimizer for Multivariate Microaggregation, whose goal is to find the appropriate attribute partitioning in order to obtain a protected data set which guarantees both utility and privacy. With this purpose, we propose a way to encode any possible solution to the problem along with some genetic operations to manipulate them. In addition, we propose an efficient measure to evaluate the quality of these solution representations.

The genetic optimizers belong to the class of Evolutionary Algorithms, which have proven efficient for different common problems. There are several necessary aspects to make the use of this type of algorithms suitable: (i) the search space must be complex and it must not be well-known, (ii) it must be possible to find a suitable encoding to represent the solutions of the problem to be optimized, and finally, (iii) it must be possible to evaluate each solution using a fitness function. In the multivariate microaggregation case, it is not clear whether there is a proper way to group the different attributes in the data set and the heuristics proposed in the literature have been shown to be suboptimal in many scenarios.

The procedure of any evolutionary algorithm described in Algorithm 1 works as follows: in general terms, a collection of instances of the alternative solutions to the problem to be optimized suffers a set of transformations to generate new instances. These are evaluated through a fitness function and they are discarded if necessary, so that after several iterations the elements that have survived to the selection process represent near-optimal solutions to the problem. This collection of elements is usually called *population*, the instances are called *chro-*

Algorithm 1: GOMM basic pseudocode

```
1 begin
2     Population $P$, $P1$, $P2$
3     while stop criterion is not met do
4         $P1 \leftarrow$ applyCrossoverOperations($P$)
5         $P2 \leftarrow$ applyCrossoverOperations($P$)
6         $P \leftarrow P \cup P1 \cup P2$
7         $P2 \leftarrow$ applySelectionOperation($P$)
8 end
```

mosomes and each iteration of the algorithm is called *generation*. The initial population is usually generated at random from all the possible solutions to the problem. Every generation, new members are created by using *crossover operations*, which combine properties of the existing members of the population, and *mutation operations*, which introduce new properties to the population by transforming a single individual chosen at random. In order to keep the number of members constant, a *selection* method is used, which in general chooses the best-fitted members to survive for the next generation. This process is repeated iteratively until a *stop condition* is found and the best member of the current population is taken as the solution of the problem.

Finding the optimal grouping configuration to cluster the different attributes in multivariate microaggregation can be seen as a grouping problem, and genetic algorithms have been proven to be a good alternative to solve this type of problems [9]. Also, it is possible to directly evaluate the suitability of a solution since a broad variety of measures for scoring the quality of an anonymization method have been proposed in the literature. Following, we explain in more detail the different components which conform GOMM.

3.1 Encoding

First, we propose a way to map an attribute grouping in a chromosome. GOMM uses the same encoding scheme as the Grouping Genetic Algorithm (GGA) [9]. Every chromosome c contains an object part c_{op}, where each gene represents the membership of the corresponding attribute in a group, and a group part c_{gp}, which contains the groups. For example, the chromosome shown in Figure 3.a) represents a solution where the attributes 1, 2 and 5 are grouped together and attributes 3 and 4 are in another group, with the group part written after the thick line. The goal of this encoding is to facilitate the work of the operators, which treat groups rather than objects.

It is important to remark that this encoding allows the presence of *clones*, which are chromosomes that represent the same attribute grouping and have the same associated fitness. For instance, Figure 3.b) represents a chromosome which maps exactly the same solution as the chromosome shown in Figure 3.a).

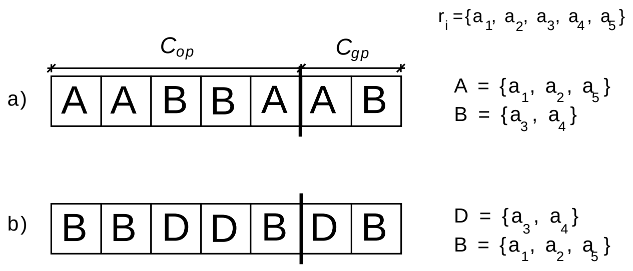


Figure 3: Parts of a simple chromosome and example of a clone

3.2 The Fitness Function

After encoding a solution in a chromosome c_i, a genetic algorithm needs a fitness function $F(c_i)$ to evaluate its quality. Chromosomes in our case represent attribute groupings that will generate a protected data set after applying microaggregation with the MDAV algorithm. It is well-know that a good anonymization method is the one that minimizes the trade-off between information loss and disclosure risk. Usually, this trade-off is computed as the arithmetic mean of both measures. Therefore, we define our fitness function as follows

$$F(c_i) = \frac{IL + DR}{2}, \qquad (2)$$

where IL stands for information loss measure and DR stands for disclosure risk measure. This function is called score and it was defined in [5] and it has been used in other evolutionary algorithms, as for instance in [11, 13]. However, because of the frequent use of the fitness function during the evolutionary process, we cannot use such a high computationally intensive function. Because of this, we will use a lighter version of the score for reducing the complexity of the original measures in our fitness function $F(c_i)$. For the sake of generality we have given the same importance to IL and DR, however, other configurations can be also valid depending on the target scenario.

3.2.1 Information Loss

Several proposals have been used in the literature in order to calculate the information loss. Depending on the alternatives, they take into account several general parameters of the data distribution such as the average vectors, covariance matrices, variance vectors or correlation matrices. Since the goal of microaggregation is to minimize the SSE, which is a specific information loss measure for any k-anonymity model, we use it as the IL measure in this work. However, the SSE itself is not suitable for the $F(c_i)$ formulation since it is not upper bounded. The most common way to normalize the SSE range into the (0..1) interval is to divide it by the Sum of Squares Total (SST) of the original data set R. This approach is also used in other works [4, 7]. The SST is defined as

$$SST = \sum_{i=1}^{n} \sum_{r_j \in n_i} (r_j - \bar{r})^T (r_j - \bar{r}), \tag{3}$$

where $\bar{r}$ is the centroid of all the original records. Note that, the SST must be computed only once during the execution since its value is constant. Then, the IL component in the fitness function of our genetic algorithm is computed as

$$IL = \frac{SSE}{SST} \cdot 100, \tag{4}$$

in this way, IL rangs between 0 and 100.

3.2.2 Disclosure Risk

In order to compute DR, usually two approaches are considered. The first one is the entity disclosure risk, which considers the scenario where intruders have access to an external data set containing a subset of non-confidential quasi-identifiers a_{nc} of the original records in R. Then, they try to link them with the corresponding protected record $r' \in R'$. Usually, the intruders apply record linkage methods [25] for this purpose. Basically, there are two families of such methods, on the one hand, those based on (in)conditional probabilities and, on the other hand, those based on distances calculations. The former (probability based family) is too inefficient to be used inside a cost function of a genetic algorithm. For this reason, we only use distance-based methods as it was done in [21] for similar reasons. This is not a problem since, as it was shown in [6], distance-based methods outperform probabilistic ones for numerical attributes. Generally, it is assumed that the intruder has several different sets of non-confidential quasi-identifiers and the risk is computed as the average risk of all those sets. However, in order to reduce the execution time of the record linkage process, we only consider the worst scenario, *i.e.* when the intruder has access to all quasi-identifiers a_{nc} of all records r stored in R. Then, they are linked with the protected data set R'. In this work, we use the Euclidean distance as in [20]. In this case, if the closest record to a known record r in R is the corresponding one r' in R', we assume that the intruder is able to find the correct link, and then, he is able to breach the privacy of such data owner. Again, as we are interested in a value fitted between $[0, 100]$, Distance Linkage disclosure (DLD) is computed as

$$DLD = \frac{links}{|R|} \cdot 100, \tag{5}$$

where $links$ is the total number of correct links achieved and $|R|$ is the number of records of the data set R.

The second considered disclosure risk scenario is the Interval Disclosure Risk (ID) which is the average percentage of protected values falling into an interval around their corresponding original values. This measure was introduced in [2]. Usually, the interval length is a user parameter. In our case, the interval is defined as $[(r_{ij} - r_{ij} \cdot 10\%), (r_{ij} + r_{ij} \cdot 10\%)]$ as in [2].

Finally, the overall DR is computed as

$$DR = (0.5ID + 0.5DLD) \tag{6}$$

3.3 Genetic Operators

Following, we describe the crossover operation and propose a new set of mutation operations and a selection operation to solve the attribute grouping problem for multivariate microaggregation.

3.3.1 Crossover

The aim of the crossover operator $\phi(c^{p1}, c^{p2})$ is to generate new members by combinig properties from different chromosomes in the current population. Two parent chromosomes c^{p1} and c^{p2} are selected randomly from the population and two new child chromosomes c^{c1} and c^{c2} are produced containing information from both parents.

Figure 4 shows the three steps of our crossover operation $\phi(c^{p1}, c^{p2})$, which is an adaptation of the Grouping Genetic Algorithm (GGA) crossover [9]. First of all, two *crossing sites* are selected from the group part of both parents (1). A crossing site c'^{p}_{gp} is a subsection of the group part of a parent chromosome c^{p}_{gp}, defined by a lower bound and an upper bound chosen at random. For example, the crossing site of the parent chromosome c^{p1} in Figure 4 is $c'^{p1}_{gp} = \{B, C\}$, with a lower bound equal to 1 and an upper bound equal to 3. Then, the groups contained in the crossing site of the first parent c^{p1} are injected into the group part of the first child c^{c1}, along with the attributes that belong to those groups (2). The rest of the attributes are grouped as they were in the second parent c^{p2} (3), so the resulting child c^{c1} has inherited properties from both parents. For the second child c^{c2}, the process is repeated exchanging the role of the parents.

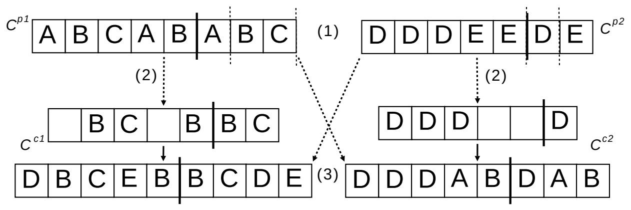


Figure 4: Graphical description of a crossover operation steps

3.3.2 Mutations

As crossover operations only combine existing properties from the current population, a way to introduce new information is needed in order to explore the whole search space; *i.e.* ensuring that any possible chromosome in the search space can be generated. Mutation operations $\varphi(c^p)$ proceed by performing random modifications to a chromosome c^p, thus generating a new member c^c with some characteristics not present in the current population.

We propose five different mutations in GOMM, which are represented in Figure 5; three working with the Group Part of the chromosome, and two with the Object Part. The former allow the exploration of the entire solution space as they deal with groups, while the latter work in a finer grane and permit to polish the solution when a promising zone in the search space has been found.

- **Group Create**($c^c = \varphi_{GC}(c^p)$) builds a new group of attributes randomly, which is injected in the child c^c. The rest of attributes are grouped as they were in the parent c^p. It is the most aggressive of all the mutations. In an extreme case a single φ_{GC} operation can transform any chromosome c^p to the one that contains only one group. The goal of φ_{GC} is to make radical transformations to the members in the population to introduce diversity. However, it is in general very disruptive and may cause the new chromosomes to contain lethal traits that lead them to the immediate elimination.

- **Group Eliminate**($c^c = \varphi_{GE}(c^p)$) selects randomly a group from the Group Part of the parent c^p and distributes all its attributes between the remaning groups, also randomly. As a result, the child c^c has a group less than its parent c^p, except in the base case that the parent has all the attributes in a single group, then a clone is generated.

- **Group Split**($c^c = \varphi_{GS}(c^p)$) choses a group at random and splits it into two different groups with the same number of attributes (if possible). The resulting child c^c has a group more than its parent c^p, except in the cases where the group that is splitted is a singleton.

- **Element Swap**($c^c = \varphi_{ES}(c^p)$) works with the Object Part of the chromosome; two attributes are selected at random and their groups are swapped, so the Group Part of the child c^c is identical to that of the parent c^p.

- **Element Move**($c^c = \varphi_{EM}(c^p)$) selects randomly an attribute from the Object Part of the parent c^p and moves it to a different existing group. The resulting child c^c has the same number of groups than his parent c^p or a group less if the element moved belonged to a singleton.

3.3.3 Selection

In order to preserve the number of members in the population and favor evolution, at the end of every iteration the chromosomes that will survive to the

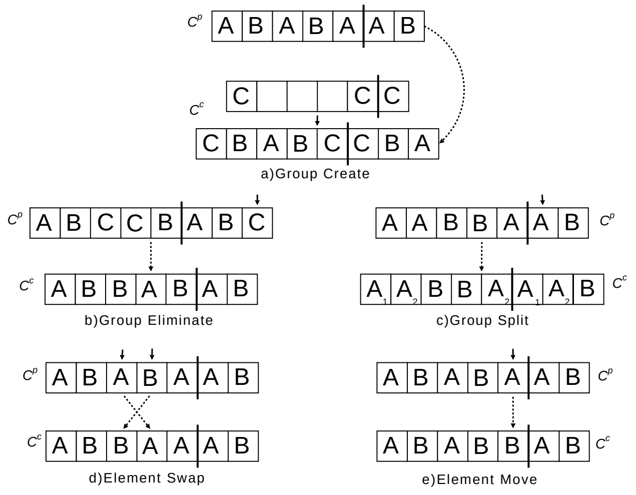


Figure 5: Steps of GOMM's mutation operations

next generation are chosen. The selection method used by GOMM is elitist as it chooses the chromosomes with a highest quality; *i.e.* those solutions with the best fitness evaluations.

Formally, if we have two chromosomes c^1 and c^2 with an associated fitness $F(c^1)$ and $F(c^2)$ and $F(c^1) < F(c^2)$, then c^1 is more likely to survive to the next generation. Thus, the objective of our genetic optimizer is to minimize the cost function $F(c)$ and obtain a solution with both low information loss and disclosure risk.

4 Genetic Operations Analysis

Now we perform a comprehensive analysis of the genetic operations used by GOMM, using the same analysis design as that proposed in [14]. The aim is to evaluate the effect that each operation has over the population for each generation of the execution. Four aspects have been studied: the number of chromosomes discarded without being used, the average chromosome life time and the efficacy and efficiency of the different operations. As a result of this analysis, we are able to understand the behavior of the genetic optimizer and to study new techniques to improve the its performance. In this section we also propose D-GOMM, an improved version of GOMM that dynamically adapts the number of genetic operations to accelerate the whole optimization process.

This analysis has been performed with real data extracted from two data sets available in the Internet. The first one if the Water-treatment data set, extracted from the UCI repository [16], which contains 38 attributes and 380 records. The second one, called Census, was extracted using the Data Extraction

System of the U.S. Census Bureau [26], which contains 12 attributes and 1080 records. A complete description about the details of the construction of this data set can be found in [8].

4.1 Analysis measures

The first way to analyze the capability of a genetic operator to introduce good properties into the population is to study the number of members discarded without being used. That is, the chromosomes that are not used to generate new chromosomes in a crossover or a mutation operation. This measure is called *utilization.* If a genetic operation introduces interesting configurations in the population, the new members generated by this operation have a higher probability to survive and to be chosen on future generations.

Another way to evaluate the effect of a genetic operator is to analyze the *average life time* of the new generated chromosomes it produces, which is the number of generations they survive. On the one hand, longer life times imply a higher probability for a given chromosome to be used in the next generations. On the other hand, an average life time of zero indicates that the chromosomes generated by the operation have not passed the selection process.

The utilization and the average life time of the chromosomes give us a first impression of the amount of useful work that a genetic operation is producing during all the algorithm execution. However, these two approaches do not show if the operations are really introducing good properties into the population; that is, if the new generated chromosomes have a better fitness than their parents, in the case of mutation operations; or a better fitness than the average of both parents, in crossover operations. This measure is called *efficacy.*

Finally, even though the previous analysis provide us with an approximate picture of the behavior of genetic operations, it does not directly reveal how much better or worse is the fitness of the new generated chromosomes. For this reason, we evaluate the *efficiency* of the operations by immediately calculating the percentage of improvement or worsement of the chromosome fitnesses after the application of the genetic operation.

To calculate the average percentage of maximum improvement and worsening for mutation operations we use Formula (7), where c^c is the child chromosome and c^p the parent chromosome. For crossover operations we use (8) and (9) where c^{p1} and c^{p2} are the parents. The idea behind the equation for the crossover operation is to calculate whether the new chromosome is better than the average fitness between both parents.

$$\%_M = \begin{cases} \frac{F(c^c)}{F(c^p)} \cdot 100 & \text{if } F(c^c) \leq F(c^p) \\ -\frac{F(c^p)}{F(c^c)} \cdot 100 & \text{if } F(c^c) > F(c^p) \end{cases} \tag{7}$$

$$r_{imp} = \frac{2 \cdot F(c^c)}{F(c^{p1}) + F(c^{p2})} \tag{8}$$

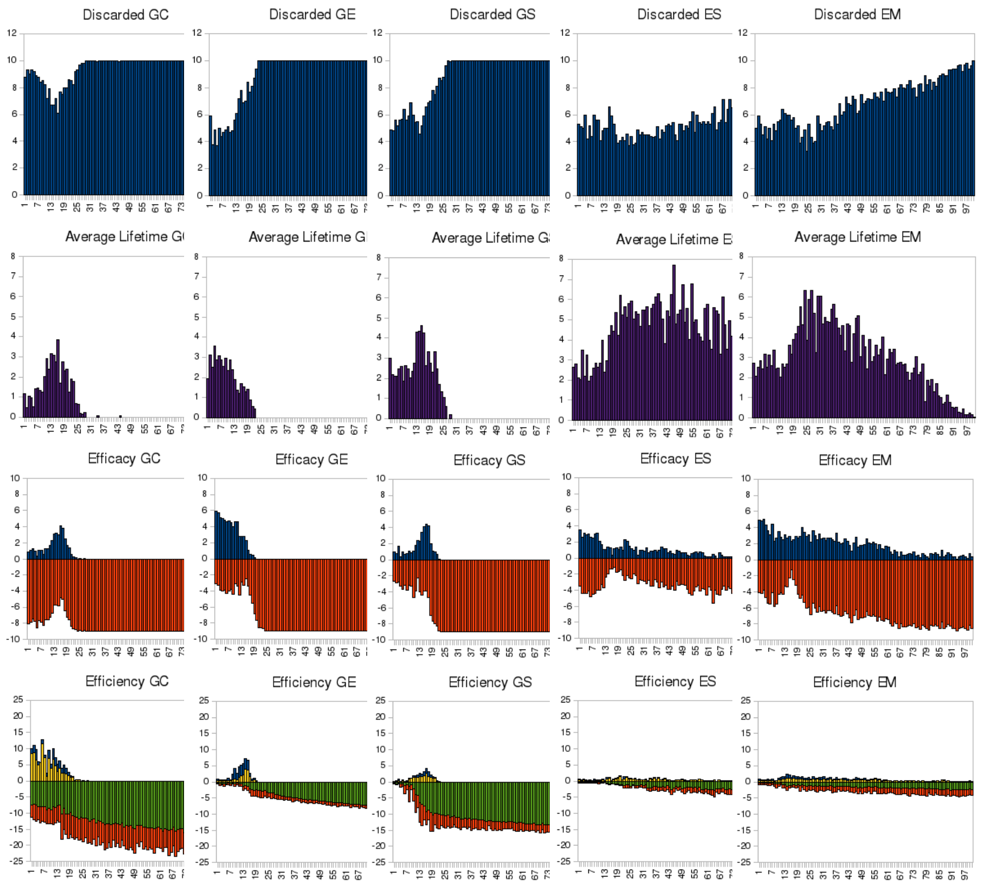


Figure 6: Average mutation operation behavior with 25 crossovers and 50 mutations per generation.

$$\%_C = \begin{cases} (1 - r_{imp}) \cdot 100 & \text{if } r_{imp} \leq 1 \\ (\frac{1}{r_{imp}} - 1) \cdot 100 & \text{if } r_{imp} > 1 \end{cases} \tag{9}$$

4.2 Analysis Results

As we said before, this analysis has been performed for two public data sets. In this subsection we only present the analysis results obtained by executing GOMM with the Water-treatment data set since the results obtained for the Census data set are very similar. We executed the optimizer with different values for the parameter k of the MDAV algorithm, but we only present the results for clusters for $k = 25$, which is a very common configuration in multivariate microaggregation since it offers a good trade-off between information loss and disclosure risk. The algorithm was executed during 100 generations, using 200 members for the population and generating 100 new members for every itera-

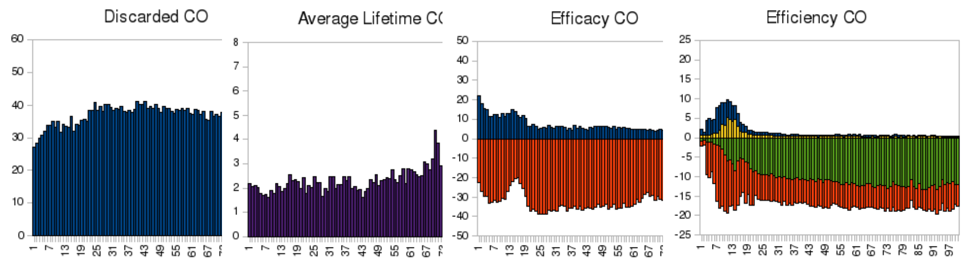


Figure 7: Average crossover operation behavior with 25 crossovers and 50 mutations per generation.

tion. These have been obtained by performing 25 crossover operations and 50 mutation operations (10 of each type) per generation.

In order to fully understand the results provided by this analysis we monitored the evolution of the population during the optimizer execution. We observed that, for a k value of 25 the average number of attribute groups of the members of the population quickly decreases. Thus, in this case, solutions with a lower number of groups are preferred to the ones with a higher number of attribute partitions.

Utilization. The first row of plots in Figure 6 shows the average number of chromosomes discarded without being used in a later operation for each mutation operation. Note that φ_{GC}, φ_{GE} and φ_{GS} are doing useless work more or less after generation 25 since 100% of the generated chromosomes are discarded without being used. In contrast, φ_{ES} and φ_{EM} gradually lose potential. Their best behavior is shown between generations 20 and 30, where less than half of the chromosomes produced are discarded without being used.

The results for the crossover operations are shown on the first plot of Figure 7. The percentage of chromosomes discarded without being used is always around 75%, being a little lower in the first generations.

Average life time. The information on the average life time for each mutation operation is shown in the second row of Figure 6. Each plot shows the average number of survived generations of the attribute grouping configurations created by the considered mutation for every iteration of the optimizer. Again, the plots show that φ_{GC}, φ_{GE} and φ_{GS} carry out unnecessary operations after generation 25 as the average life time of all the chromosomes created by these operations is zero, that is, they die as soon as they are created. On the other hand, the other two mutations improve they performance after generation 20 since the chromosomes they create live longer. The average life time of the chromosomes produced by φ_{EM} does not decrease until generation 50, and in the case of φ_{ES} until generation 70.

Regarding crossover operations, we can see in the second plot of Figure 7 that the average life time is always around 2 and 3, which shows the conservative behaviour of this operation.

Efficacy. The efficacy plots presented in the third row of figure 6 show the number of improved chromosomes along with the number of worsened ones for every generation. It seems reasonable to assume that the addition of the two parts should be the number chromosomes generated for every operation in a certain generation (10 for each mutation and 50 for the crossover); nevertheless, this is not true when the generated chromosome is a clone of its parent, that is, they have the same fitness.

Again, φ_{GC}, φ_{GE} ang φ_{GC} only generate chromosomes worse that those of their parents from generation 25. However, the only operation that gradually loses efficacy is φ_{GE}, as φ_{GC} and φ_{GS} show a peak near generations 18-19. This is because the average number of groups of the first population is large because it has been generated randomly, and thus, creating small groups or splitting them does no generate better chromosomes. However, as the number of groups decreases due to the creation of large groups and the elimination of others, these operations are able to generate better chromosomes. In the case of φ_{ES}, the plot shows this is an operation that generates a lot of clones, but until the end of the execution is capable of generating improved chromosomes. This also happens with φ_{EM}, where we can observe that near generations 16-17 the number of clones generated increases. The only case where φ_{EM} produces clones is when the parent chromosome has a single group, which is exactly when φ_{GS} and φ_{GC} start generating improved chromosomes.

The crossover results in the third plot of Figure 7 show that after generation 20 the number of improvements is very constant, standing in the 5% of the chromosomes generated.

Efficiency. In the last row of Figure 6, we can see the average efficiency results for each mutation operation, showing the average improvement and the average worsening along with the average maximum and minimum for every generation. We observe that φ_{GC} and φ_{GE} introduce very interesting properties in the first generations, with average maximum efficiency values around 10%. This is caused by the fact that, as we said before, the members of the first population have an elevated number of groups, and as these mutations reduce it, they are able to generate much better chromosomes. Between generation 10 and 20 φ_{GS} does some useful work because new chromosomes with a low number of groups have appeared in the population due to the effects of φ_{GC} and φ_{GE}. These three operations, as seen before, stop doing useful work after generation 25 as they only produce worse chromosomes. In the case of φ_{ES} and φ_{EM}, we notice that their efficiency values are lower than the other mutation operations, standing around 2% in their phase of best behavior. This happens because these operations work with attributes rather than groups, so the chromosomes they generate are very similar to the parent chromosome they used. That property is the one that allows φ_{ES} and φ_{EM} to continue doing useful work after generation 25, when the optimum number of groups has been found and φ_{GC}, φ_{GE} and φ_{GS} do useless work.

The efficiency results for the crossover operation are presented in the last plot of Figure 7. In the first generations, the average maximum efficiency shows a peak and afterwards it quickly decreases, showing that the new chromosomes are worse or have the same fitness as the parents. The explanation to this phenomenon is that, in the first generations, parent chromosomes with a large number of groups are crossed with others with few groups, so the resulting children have less groups than the former parents. In the next generations, when the number of groups of the members of the population has stabilized, the new chromosomes generated are very similar to their parents or have more groups, which means that they have the same or worse fitness respectively.

4.3 Improving performance

The main conclusion of the previous analysis is that some operations carry out a lot of useless work after a given generation. As the algorithm proceeds we distinguish two separated phases: firstly, the group-oriented mutations (φ_{GC} , φ_{GE} and φ_{GS}) contribute to evolve the population modifying the number of groups; secondly, these operators only generate worse chromosomes and the object-oriented mutations, along with the crossover, introduce improvements in a lower scale.

In this section we propose D-GOMM, the Dynamic Genetic Optimizer for Multivariate Microaggregation, which includes a control mechanism to detect the generation where the second phase starts. If the optimizer is able to detect when an operator is performing useless work, that is, generating chromosomes worse-fitted than their parents, it can stop its execution. This means that less child chromosomes will be generated, thus performing less fitness evaluations and accelerating the optimization process.

It is important to notice that D-GOMM only stops the execution of the genetic operations that perform unnecessary work during several generations. Thus, for all the executions performed the quality of the solution given by D-GOMM is always equal to the one resulting of GOMMs execution.

5 Experiments

In this section we study the solutions given by our genetic optimizer from a twofold perspective. Firstly, we validate the solutions provided by GOMM according to the microaggregation problem. Secondly, we compare them with the ad-hoc solutions used in practice. To do that, we have executed GOMM with different values of k (from 5 to 100) to study the impact of such parameter in the configuration of the groups in the best solution. In addition, we also show in this section a performance comparison between GOMM and D-GOMM.

Solution validation. Intuitively, when k increases, the clusters size must grow producing an increment in the information loss of each cluster. This effect combined with the problem of dimensionality make configurations with a large number of groups to be preferred to configurations with only one or two groups, as the values of k increase.

Tables 1 and 2 show the average solution given by GOMM using the Water Treatment data set and the Census data set, respectively. The value of parameter k is placed in the first column and the average number of groups of the solution in the last one. The other columns show the average values of the cost function parameters.

Water-treatment

k	IL	ID	DLD	*score*	*groups*
5	35.74	27.93	15.89	28.82	1
10	48.05	25.07	7.15	32.08	1
25	50.11	23.62	18.29	35.53	2
50	57.85	23.23	6.91	36.46	2
100	63.38	21.77	14.18	40.68	5

Table 1: Fitness (score) and number of groups of the best configuration found by GOMM with different parameterizations using the Water-treatment data set.

Census

k	IL	ID	DLD	*score*	*groups*
5	8.01	32.01	14.56	15.64	1
10	11.97	25.90	6.69	14.13	1
25	17.97	21.06	2.53	14.88	1
50	18.36	18.15	6.02	15.23	2
100	16.91	21.85	5.57	15.31	2

Table 2: Fitness (score) and number of groups of the best configuration found by GOMM with different parameterizations using the Census data set.

When k increases, the number of groups of the best solution also increases, specially due to an increment of the IL component, as our intuition predicted. Therefore, the solutions yielded by GOMM are coherent with our intuition.

From these tables we can also observe that finding the optimal value for parameter k is far from trivial. For instance, in the Census table, the best scores are presented by configurations with a value k between 10 and 50.

Solution comparison. In order to compare our results with the different strategies presented in the literature, we manually split both data sets following the recommendations described in [18]. The resulting configurations are depicted in Tables 3 and 4. As we can observe, configurations vary from few groups containing many attributes to many groups containing a few attributes. In such configurations, we also considered the correlation between attributes. On the one hand, we grouped correlated attributes together (to minimize the information loss). On the other hand, we grouped correlated attributes in different groups (to minimize the disclosure risk). Finally, we also consider two special scenarios: in the first one, all attributes are grouped together, ensuring k anonymity. In the second one, we consider the univariate microaggregation scenario where each attribute is microaggregated separately.

The results presented in Tables 5 and 6 show the fitness value of all the ad-hoc configurations along with the fitness value obtained by GOMM for different values of k. If we compare the results obtained using the same k value,

Water treatment	
G_c^{10}	$(a_1, a_4, a_5, a_6, a_7, a_{11}, a_{13}, a_{17}, a_{20}, a_{36})$ $(a_3, a_9, a_{10}, a_{15}, a_{16}, a_{18}, a_{21}, a_{22}, a_{23}, a_{29})$ $(a_2, a_8, a_{12}, a_{14}, a_{19}, a_{26}, a_{28}, a_{31}, a_{33}, a_{37})$ $(a_{24}, a_{25}, a_{27}, a_{30}, a_{32}, a_{34}, a_{35}, a_{38})$
G_{nc}^{10}	$(a_1, a_2, a_3, a_{24}, a_4, a_9, a_8, a_{25}, a_5, a_{10})$ $(a_{12}, a_{27}, a_6, a_{15}, a_{14}, a_{30}, a_7, a_{16}, a_{19}, a_{32})$ $(a_{11}, a_{18}, a_{26}, a_{34}, a_{13}, a_{21}, a_{28}, a_{35}, a_{17}, a_{22})$ $(a_{20}, a_{36}, a_{29}, a_{23}, a_{31}, a_{33}, a_{37}, a_{38})$
G_c^5	$(a_1, a_4, a_5, a_6, a_7)(a_{11}, a_{13}, a_{17}, a_{20}, a_{36})$ $(a_3, a_9, a_{10}, a_{15}, a_{16})(a_{18}, a_{21}, a_{22}, a_{23}, a_{29})$ $(a_2, a_8, a_{12}, a_{14}, a_{19})(a_{26}, a_{28}, a_{31}, a_{33}, a_{37})$ $(a_{24}, a_{25}, a_{27}, a_{30})(a_{32}, a_{34}, a_{35}, a_{38})$
G_{nc}^5	$(a_1, a_2, a_3, a_{24}, a_4)(a_9, a_8, a_{25}, a_5, a_{10})$ $(a_{12}, a_{27}, a_6, a_{15}, a_{14}, a_{30})(a_7, a_{16}, a_{19}, a_{32})$ $(a_{11}, a_{18}, a_{26}, a_{34}, a_{13}, a_{21})(a_{28}, a_{35}, a_{17}, a_{22})$ $(a_{20}, a_{36}, a_{29}, a_{23})(a_{31}, a_{33}, a_{37}, a_{38})$
G_c^3	$(a_1, a_4, a_5)(a_6, a_7, a_{11})(a_{13}, a_{17}, a_{20})(a_3, a_9, a_{10})$ $(a_{15}, a_{16}, a_{18})(a_{21}, a_{22}, a_{23})(a_2, a_8, a_{12})(a_{14}, a_{19}, a_{26})$ $(a_{28}, a_{31}, a_{33})(a_{29}, a_{36}, a_{37})$ $(a_{24}, a_{25}, a_{27})(a_{30}, a_{32}, a_{34})(a_{35}, a_{38})$
G_{nc}^3	$(a_1, a_2, a_3)(a_{24}, a_4, a_9)(a_8, a_{25}, a_5)(a_{12}, a_{27}, a_6)$ $(a_{15}, a_{14}, a_{30})(a_7, a_{16}, a_{19})(a_{11}, a_{18}, a_{26})(a_{34}, a_{13}, a_{21})$ $(a_{28}, a_{35}, a_{17})(a_{10}, a_{32}, a_{22})$ $(a_{20}, a_{36}, a_{29})(a_{23}, a_{31}, a_{33})(a_{37}, a_{38})$

Table 3: Ad-hoc group configurations for the Water treatment data set. G_c^n stands for groups of n correlated attributes and G_{nc}^n stands for groups of n non-correlated attributes.

we observe that GOMM has always the best quality, unless the cases where GOMM yields a solution containing a single group, which is one of the ad-hoc groupings we executed. For instance, if we compare the Water treatment results (Table 5) obtained with $k = 50$, we see that GOMM achieves a fitness equal to 34.3 splitting the data set in three different groups, whilst the best ad-hoc configuration only achieves a fitness value equal to 40.9 dividing the data set into four groups. Similar results are obtained with the remaining k values.

Also, if we compare GOMM results with the Census data set, we observe very similar results. For example, with $k = 25$ GOMM achieves a fitness value equal to 13.0 dividing the Census data set into two groups and the best ad-hoc solution is equal to 14.3 without splitting the data set.

Performance results. The aim of this last section is to compare the two versions of our genetic approach in terms of execution time. In Section 4.3 we introduced D-GOMM, which is a version of the GOMM algorithm that detects when a genetic operation is producing useless work and in this case, it stops exe-

Census

G_c^4	$(a_2, a_4, a_6, a_7)(a_5, a_{10}, a_{11}, a_{12})$ (a_1, a_3, a_8, a_9)
G_{nc}^4	$(a_1, a_2, a_4, a_5)(a_3, a_6, a_{10}, a_{11})$ (a_7, a_8, a_9, a_{12})
G_c^3	$(a_2, a_4, a_7)(a_3, a_{11}, a_{12})$ $(a_5, a_6, a_{10})(a_1, a_8, a_9)$
G_{nc}^3	$(a_2, a_3, a_5)(a_8, a_{10}, a_{11})$ $(a_7, a_9, a_{12})(a_1, a_4, a_6)$

Table 4: Ad-hoc group configuration for the census data set. G_c^n stands for groups of n correlated attributes and G_{nc}^n stands for groups of n non-correlated attributes.

Water-treatment

	k=5	k=10	k=25	k=50	k=100
G^{38}	28.8 (1)	32.1 (1)	36.9 (1)	42.0 (1)	50.6 (1)
G^1	49.4 (38)	48.9 (38)	47.2 (38)	45.0 (38)	46.2 (38)
G_c^{10}	43.8 (4)	45.4 (4)	44.6 (4)	42.7 (4)	42.3 (4)
G_{nc}^{10}	43.6 (4)	46.2 (4)	44.7 (4)	42.7 (4)	45.5 (4)
G_c^5	42.1 (8)	44.0 (8)	47.0 (8)	49.7 (8)	47.0 (8)
G_{nc}^5	42.8 (8)	45.3 (8)	48.7 (8)	49.9 (8)	50.3 (8)
G_c^3	42.4 (13)	43.2 (13)	46.7 (13)	50.1 (13)	52.7 (13)
G_{nc}^3	42.3 (13)	43.4 (13)	46.1 (13)	49.8 (13)	52.3 (13)
$GOMM$	28.8 (1)	32.1 (1)	35.5 (2)	36.7 (2)	40.7 (5)

Table 5: Fitness (score) of some ad-hoc grouping and the solution given by GOMM using the Census data set with different values of the k parameter. Number of groups are depicted in parenthesis.

cuting it. Therefore, the number of new generated chromosomes per generation decreases along with the number of fitness evaluations of this offspring.

The plot presented in Figure 8 shows the execution time comparison between GOMM and D-GOMM using a data set with 38 attributes. Notice that for different values of the k parameter D-GOMM is always faster than GOMM. However, the speed-up achieved varies depending on the value of the k parameter. We computed the speed-ups in all the scenarios and the results show that as k increases, the speed-up decreases. The explanation to this phenomenon is that when k is small, the solution given by the algorithm has less groups of attributes, thus, the optimizer converges faster and the group-oriented mutations stop doing useful work before.

Census

	k=5	k=10	k=25	k=50	k=100
G^{12}	15.6 (1)	14.1 (1)	14.9 (1)	15.9 (1)	18.1 (1)
G^{1}	48.9 (12)	48.1 (12)	45.3 (12)	41.8 (12)	37.6 (12)
G_c^4	39.7 (3)	37.4 (3)	32.1 (3)	26.4 (3)	19.8 (3)
G_{nc}^4	34.7 (3)	28.5 (3)	21.8 (3)	17.2 (3)	16.5 (3)
G_c^3	40.4 (4)	38.1 (4)	33.3 (4)	27.7 (4)	22.3 (4)
G_{nc}^3	38.0 (4)	34.5 (4)	28.1 (4)	22.5 (4)	18.3 (4)
$GOMM$	15.6 (1)	14.1 (1)	14.9 (1)	15.2 (2)	15.3 (2)

Table 6: Fitness (score) of some ad-hoc grouping and the solution given by GOMM using the Census data set with different values of the k parameter. Number of groups are depicted in parenthesis.

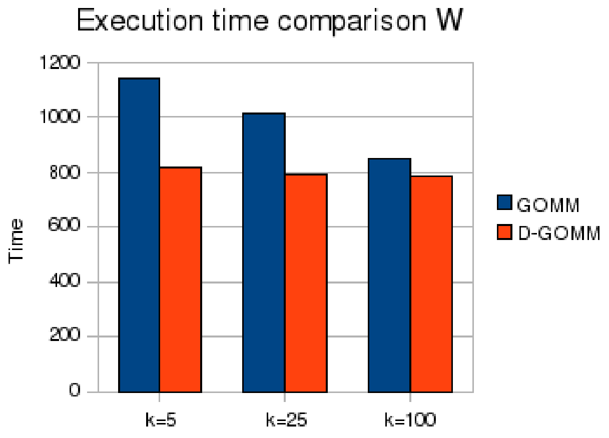


Figure 8: Execution time of GOMM and D-GOMM for different values of the k parameter

6 Conclusions

In this paper, we have proposed GOMM, a genetic optimizer to find the best way to group the attributes when anonymizing a dataset with multivariate microaggregation, in order to obtain a protected dataset with both low disclosure risk and information loss. The experiments performed show that the solutions given by GOMM outperform other grouping strategies proposed in the literature, for different parametrizations of the multivariate microaggregation method.

In addition, we analyzed the behavior of our optimizer in order to understand the contribution of every genetic operation in the solution given. This analysis allows the study of alternatives to improve GOMM's performance. One of these has been proposed in a version called D-GOMM, which dynamically detects when a genetic operation is doing useless work and then stops its execution. We have seen that this mechanism improves the execution time of our optimizer, while the quality of the solution given is not altered.

Regarding future work, we will study other fitness functions for GOMM based on convex optimization approaches. Since IL and DR are contradictory goals convex minimization seems a very promising research line. Apart from that, we consider two possible directions to follow to improve GOMM performance. The first one is to use other multivariate microaggregation methods faster than the MDAV algorithm, since we observed that nearly 60% of the optimizer execution time is spent in this function. Secondly, we want to study parallelization strategies of our approach because there are several points in GOMM where parallelism could be opened. Both future lines aim at improving GOMM's performance and specially, at allowing the optimizer to deal with larger data sets.

Acknowledgements

Partial support by the Spanish MEC (projects ARES – CONSOLIDER INGENIO 2010 CSD2007-00004 – and eAEGIS – TSI2007-65406-C03-02) is acknowledged.

References

[1] C. Aggarwal. On k-anonymity and the curse of dimensionality. In *Proc. of the 31st Int. Conf. on Very Large Databases*, pages 901–909, 2005.

[2] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 439–450, 2000.

[3] T. Dalenius and S. Reiss. Data-swapping: a technique for disclosure control. *Journal of Statistical Planning and Inference*, 6:73–85, 1982.

[4] J. Domingo-Ferrer and J. M. Mateo-Sanz. Practical data-oriented microaggregation for statistical disclosure control. *IEEE Trans. on Knowledge and Data Engineering*, 14(1):189–201, 2002.

[5] J. Domingo-Ferrer and V. Torra. Disclosure control methods and information loss for microdata. In *Confidentiality, disclosure, and data access : Theory and practical applications for statistical agencies*, pages 91–110. Elsevier, 2001.

[6] J. Domingo-Ferrer and V. Torra. Validating distance-based record linkage with probabilistic record linkage. In *Proc. of the Catalan Conference on Artificial Intelligence*, pages 207–215, 2002.

[7] J. Domingo-Ferrer and V. Torra. Ordinal, continuous and heterogeneous k-anonymity through microaggregation. *Data Mining and Knowledge Discovery*, 11(195–212), 2005.

[8] J. Domingo-Ferrer, V. Torra, J. M. Mateo-Sanz, and F. Sebé. Systematic measures of re-identification risk based on the probabilistic links of the partially synthetic data back to the original microdata. Technical report, 2005.

[9] E. Falkenauer. *Genetic Algorithms and Grouping Problems*. John Wiley & Sons, Inc., New York, NY, USA, 1998.

[10] A. Hundepool, J. Domingo-Ferrer, L. Franconi, S. Giessing, E. S. Nordholt, K. Spicer, and P.-P. de Wolf. *Statistical Disclosure Control*. Wiley, 2012.

[11] J. Jimenez, J. Marés, and V. Torra. An evolutionary approach to enhance data privacy. *Soft Computing*, 15(7):1301–1311, 2011.

[12] J. Lane, P. Heus, and T. Mulcahy. Data access in a cyber world: making use of cyberinfrastructure. *Transactions on Data Privacy*, 1(1):2–16, 2008.

[13] J. Marés and V. Torra. An evolutionary optimization approach for categorical data protection. In *EDBT/ICDT Workshops*, pages 148–157, 2012.

[14] V. Muntés-Mulero. *Genetic Optimization for large join queries*. PhD thesis, Universitat Politècnica de Catalunya, 2007.

[15] V. Muntés-Mulero and J. Nin. Privacy and anonymization for very large datasets. In *Proc of the 18th ACM Conference on Information and Knowledge Management*, pages 2117–2118, 2009.

[16] P. Murphy and D. Aha. UCI Repository machine learning databases. *Irvine, CA: University of California, Department of Information and Computer Science*, 1994.

[17] J. Nin, J. Herranz, and V. Torra. Attribute selection in multivariate microaggregation. In *Post-Proc. of 11th ACM International Conference on Extending Database Technology (EDBT)*, pages 51–60, 2008.

[18] J. Nin, J. Herranz, and V. Torra. How to group attributes in multivariate microaggregation. *Int. J. of Unc., Fuzz. and Knowledge Based Systems*, 16(1):121–138, 2008.

[19] A. Oganian and J. Domingo-Ferer. On the complexity of optimal microaggregation for statistical disclosure control. *Statistical Journal United Nations Economic Commission for Europe*, 18(4):345–354, 2000.

[20] D. Pagliuca and G. Seri. Some results of individual ranking method on the system of enterprise accounts annual survey. Technical report, Esprit SDC Project, Deliverable MI-3/D2, 1999.

[21] F. Sebé, J. Domingo-Ferrer, J. M. Mateo-Sanz, and V. Torra. Post-masking optimization of the tradeoff between information loss and disclosure risk in masked microdata sets. In *Inference Control in Statistical Databases, Lecture Notes in Computer Science 2316*, pages 187–196. J. Domingo-Ferrer (Ed.), 2002.

[22] M. Solé, V. Muntés-Mulero, and J. Nin. Efficient microaggregation techniques for large numerical data volumes. *International Journal of Information Security (IJIS)*, 11(253-267), 2012.

[23] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *Int. J. of Unc., Fuzz. and Knowledge Based Systems*, 10(5):571–588, 2002.

[24] L. Sweeney. k-anonymity: a model for protecting privacy. *Int. J. of Unc., Fuzz. and Knowledge Based Systems*, 10(5):557–570, 2002.

[25] V. Torra and J. Domingo-Ferrer. Record linkage methods for multidatabase data mining. In *Information Fusion in Data Mining*, pages 101–132. Springer, 2003.

[26] U.S. Census Bureau, Data Extraction System, http://www.census.gov/.

[27] L. Willenborg and T. de Waal. *Elements of Statistical Diclosure Control*. Lecture Notes in Statistics. Springer, 2001.