Practical Data-Oriented Microaggregation for Statistical Disclosure Control

Josep Domingo-Ferrer, *Member*, *IEEE*, and Josep M. Mateo-Sanz

Abstract—Microaggregation is a statistical disclosure control technique for microdata disseminated in statistical databases. Raw microdata (i.e., individual records or data vectors) are grouped into small aggregates prior to publication. Each aggregate should contain at least k data vectors to prevent disclosure of individual information, where k is a constant value preset by the data protector. No exact polynomial algorithms are known to date to microaggregate optimally, i.e., with minimal variability loss. Methods in the literature rank data and partition them into groups of fixed-size; in the multivariate case, ranking is performed by projecting data vectors onto a single axis. In this paper, candidate optimal solutions to the multivariate and univariate microaggregation problems are characterized. In the univariate case, two heuristics based on hierarchical clustering and genetic algorithms are introduced which are data-oriented in that they try to preserve natural data aggregates. In the multivariate case, fixed-size and hierarchical clustering microaggregation algorithms are presented which do not require data to be projected onto a single dimension; such methods clearly reduce variability loss as compared to conventional multivariate microaggregation on projected data.

Index Terms—Statistical databases, microdata protection, statistical disclosure control, microaggregation, hierarchical clustering, genetic algorithms.

1 Introduction

Statistical offices release two kinds of data through their statistical databases: tabular data and microdata sets (individual respondent records). In both cases, data dissemination should be performed in a way that does not lead to disclosure of individual information, but preserves the informational content as much as possible (disclosure control problem). While there is long experience in table dissemination, microdata dissemination is a much more recent activity.

Following the nomenclature of [24], a *microdata set* is a set of records or *data vectors* containing data of individual respondents who can be persons, companies, etc. The individual data vectors of a microdata set are stored in a *microdata file*. Each individual j is assigned a data vector V_j formed by two kinds of variables:

Key variables: These variables are identifiers of the individual. A key variable can be a *direct identifier* (e.g., the name) or an *indirect identifier* (e.g., the phone number).

Sensitive variables: They contain sensitive information on the individual (e.g., salary).

Key variables allow identification of the individual. If a direct identifier is known by the intruder, then he can uniquely identify a specific individual without additional knowledge. Unique identification through an indirect identifier is not possible, although it may become possible if additional a priori information on the individual is available (e.g., the hair color does not uniquely identify an individual; however, if one has a data set with the hair color and the city as variables, a red-haired individual known to live in a Chinese village may be readily identified). The rest of the variables in a data vector are sensitive variables which belong to the private domain of the individuals.

Both key and sensitive variables are confidential because they contain information at the individual level. This fact means that microdata sets should be protected against disclosure before being released. In [22], a distinction between two kinds of disclosure is made. *Reidentification disclosure* occurs if the intruder is able to deduce the value of a sensitive variable for the target individual after this individual has been reidentified; approaches to assessing the reidentification risk can be found in [4] and [20]. *Prediction disclosure* occurs if the data enable to predict the value of a sensitive variable for some target individual; for prediction disclosure, it is not necessary that reidentification has taken place; see [9] for further discussion on prediction disclosure.

Disclosure control methods for microdata sets fall into three main categories (see [24], [1], [8]):

Global recoding: The granularity of the microdata set is reduced. A qualitative variable is transformed by collapsing categories into another qualitative variable with more general categories. A quantitative variable is transformed into a qualitative variable where categories are intervals defined on the range of the original variable.

Local suppression: Some information in the microdata set is suppressed prior to publication.

- *J. Domingo-Ferrer is with the Department of Computer Science, Universitat Rovira i Virgli, Autovia de Salou s/n, E-43006 Tarragona, Catalonia, Spain. E-mail: jdomingo@etse.urv.es.*
- *J.M. Mateo-Sanz is with the Statistic and OR Group, Department of Chemical Engineering, Universitat Rovira i Virgli, Autovia de Salou s/n, E-43006 Tarragona, Catalonia, Spain. E-mail: jmateo@etse.urv.es.*

Manuscript received 6 Feb. 1998; revised 1 Apr. 1999; accepted 31 Aug. 2000; posted to Digital Library 12 June 2001.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 106236.

1041-4347/02/$17.00 © 2002 IEEE

Substitution (or perturbation): The values of each variable in the microdata set are perturbed and the perturbed values are used to replace the original values.

The main problem in disclosure control is to provide sufficient protection without seriously damaging the information contained in the original data.

Section 2 introduces basic microaggregation concepts. Candidate optimal solutions to the multivariate and univariate microaggregation problems are characterized in Section 3. Heuristic methods for univariate microaggregation based on hierarchical clustering and genetic algorithms are presented and compared in Section 4. Section 5 discusses and compares methods for multivariate microggregation for both projected and unprojected data. Finally, in Section 6, conclusions are drawn and topics for future research are identified. Appendix A contains the proofs of the theoretical results stated in the paper. Appendix B contains a description of Ward's hierarchical clustering method.

2 Microaggregation Concepts

Microaggregation is a family of statistical disclosure control techniques for *quantitative* (numeric) microdata which belong to the substitution/perturbation category. The rationale behind microaggregation is that confidentiality rules in use allow publication of microdata sets if the data vectors correspond to groups of k or more individuals, where no individual dominates (i.e., contributes too much to) the group and k is a threshold value. Strict application of such confidentiality rules leads to replacing individual values with values computed on small aggregates (microaggregates) prior to publication. This is the basic principle of microaggregation.

To obtain microaggregates in a microdata set with n data vectors, these are combined to form g groups of size at least k. For each variable, the average value over each group is computed and is used to replace each of the original averaged values. Groups are formed using a criterion of maximal similarity. Once the procedure has been completed, the resulting (modified) data vectors can be published.

The partition problem implicit in microaggregation differs from the classical clustering problem whose goal is to split a population into a fixed number of disjoint groups [14], regardless of the group size. The constraint in microaggregation is on the group size, rather than on the number of groups. Partitions resulting from microaggregation cannot consist of groups of size smaller than k; call such partitions k-partitions. To solve the k-partition problem, a measure of similarity between data vectors is needed. Each individual data vector can be viewed as a point and the whole microdata set as a set of multidimensional points. The dimension is the number of variables in data vectors. If data vectors are characterized as points, the similarity between them can be measured using a distance.

To be more specific, consider a microdata set with p continuous variables and n data vectors (i.e., the result of observing p variables on n individuals). A particular data vector can be viewed as an instance of $\mathbf{X}' = (X_1, \cdots, X_p)$, where the X_i are the variables. With these individuals, g groups are formed with n_i individuals in the ith group ($n_i \geq k$ and $n = \sum_{i=1}^{g} n_i$). Denote by x_{ij} the jth data vector in the ith group; denote by $\bar{\mathbf{x}}_i$ the average data vector over the ith group and by $\bar{\mathbf{x}}$ the average data vector over the whole set of n individuals.

The optimal k-partition is defined to be the one that maximizes within-group homogeneity; the higher the within-group homogeneity, the lower the information loss since microaggregation replaces values in a group by the group centroid. The sum of squares criterion is common to measure homogeneity in clustering [23], [10], [16], [17], [12], [13]. The within-groups sum of squares SSE is defined as

$$SSE = \sum_{i=1}^{g} \sum_{j=1}^{n_i} (x_{ij} - \bar{\mathbf{x}}_i)'(x_{ij} - \bar{\mathbf{x}}_i). \quad (1)$$

The lower SSE, the higher the within-group homogeneity. The between-groups sum of squares SSA is

$$SSA = \sum_{i=1}^{g} n_i (\bar{\mathbf{x}}_i - \bar{\mathbf{x}})'(\bar{\mathbf{x}}_i - \bar{\mathbf{x}}). \quad (2)$$

In the proofs given in Appendix A, the following alternative expression for SSA in the univariate case will be used:

$$SSA = \sum_{i=1}^{g} t_i^2/n_i - t^2/n, \quad (3)$$

where t_i is the total sum of the ith group and t is the grand total of all groups. The total sum of squares is $SST = SSA + SSE$ or, explicitly,

$$SST = \sum_{i=1}^{g} \sum_{j=1}^{n_i} (\mathbf{x}_{ij} - \bar{\mathbf{x}})'(\mathbf{x}_{ij} - \bar{\mathbf{x}}). \quad (4)$$

In terms of sums of squares, the optimal k-partition is the one that minimizes SSE (or, equivalently, maximizes SSA). A measure L of information loss standardized between 0 and 1 can be obtained from

$$L = \frac{SSE}{SST}. \quad (5)$$

Remark that, if preserving variability is not equally critical for all variables in $X_1, \cdots, X_p$, then variables can be weighted in (1), (2), and (4).

Note 1 (Fixed vs. variable group size). Classical microaggregation [3], [6], [7] required that all groups, except perhaps one, be of size k; allowing groups to be of size $\geq k$, depending on the structure of data, can be termed *data-oriented microaggregation* [18]. Fig. 1 illustrates the advantages of variable-sized groups. If classical fixed-size microaggregation with $k = 3$ is used, we obtain a partition of the data into three groups, which looks rather unnatural for the data distribution given. On the other hand, if variable-sized groups are allowed, then the five data on the left can be kept in a single group and the four data on the right in another group; such a

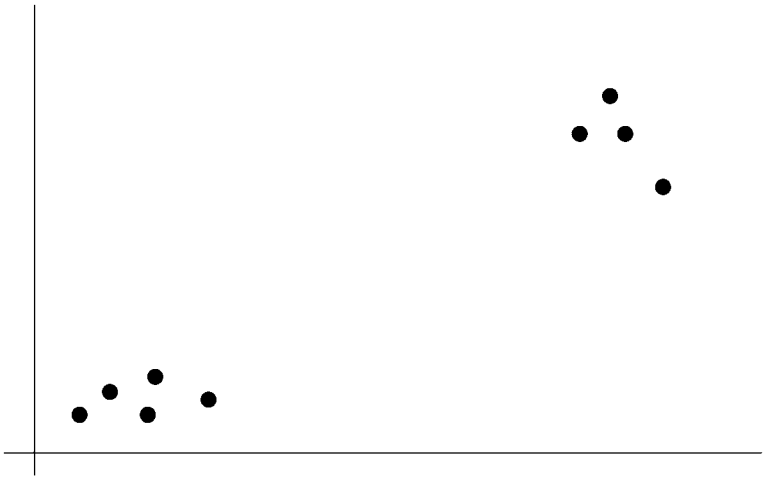

Fig. 1. Variable-sized groups versus fixed-sized groups.

variable-size grouping yields more homogeneous groups, which implies lower information loss.

3 Optimal Microaggregation

The problem of optimal microaggregation defined in Section 2 is related but not identical to the classical minimum sum-of-squares clustering (MSSC) whose goal is to find a partition of a data set into a *fixed* number of disjoint groups (without size constraints) so that the within-groups sum of squares is minimum. MSSC is known to be NP-hard [5]. To the best of our knowledge, no complexity assessment nor polynomial exact algorithms are available in the literature for optimal microaggregation, which is a relatively new problem. Therefore, heuristic methods seem the most viable approach. We next state some theoretical results which are useful to devise heuristic solutions. Proofs can be found in Appendix A.

Proposition 1. *An optimal solution to the k-partition problem of a set of data exists such that its groups have size greater than or equal to k and less than $2k$.*

Using Proposition 1, the search space for the optimal solution to the k-partition problem can be reduced to the set of candidate solutions having groups with sizes between k and $2k-1$.

Definition 1. *For a given data set, k-partition p is said to be finer than k-partition p' if every group in p is contained by a group in p'.*

It is straightforward to check that "finer than" is a partial order relationship on the set of k-partitions of a given data set.

Definition 2. *For a given data set, a k-partition p is said to be minimal with respect to the relationship "finer than" if there is no k-partition $p' \neq p$ such that p' is finer than p.*

Proposition 2. *For a given data set, a k-partition p is minimal with respect to the relationship "finer than" if and only if it consists of groups with sizes $\geq k$ and $< 2k$.*

The following corollary results from Propositions 1 and 2:

Corollary 1. *An optimal solution to the k-partition problem of a set of data exists that is minimal with respect to the relationship "finer than."*

Proposition 1 and Corollary 1 give two equivalent characterizations that greatly reduce the search space for the k-partition problem of a set of n data vectors.

3.1 Optimal Univariate Microaggregation

Univariate microaggregation refers to either of the two following cases:

Univariate data set: The microdata set being microaggregated consists of a single variable.

Individual ranking: The microdata set to be microaggregated consists of several variables, but those are considered independently in turn. Data vectors are ranked by the first variable and the values of this variable are microaggregated. Then, the same procedure is repeated for the second variable and so on.

For univariate microaggregation, further characterization of optimal solutions is possible. In what follows, we use the term "element" to denote a univariate data vector.

Definition 3 (Connected group). *Given a set S of data, a group $G \subseteq S$ is said to be connected if, for any elements x, y, z such that $x, y \in G$, $z \in S$ and $x \leq z \leq y$, it holds that $z \in G$.*

Proofs for the results below can be found in Appendix A.

Lemma 1. *Consider a set of n data, where $n \geq 2k$, to be split into two disjoint groups of size $\geq k$ each. Then, the groups in the optimal two-group k-partition are both connected.*

Lemma 2. *If the optimal k-partition for a set of data consists of g groups, then any subset of $g-1$ groups is an optimal k-partition for the corresponding subset of data.*

The following result nontrivially generalizes to an arbitrary k a result of [11] for the specific case $k=1$, that is, for the standard partition problem.

Theorem 1. *The optimal solution to the k-partition problem of a set of data consists of connected groups.*

Using Theorem 1 and Proposition 1, *the search space for the optimal solution to the k-partition problem can be reduced to the set of candidate solutions having connected groups with sizes between k and $2k-1$.* This is illustrated in Example 1.

Example 1. Let $\{d_1, d_2, \cdots, d_9\}$ be a univariate data set. Assume that the elements are ranked in ascending order, i.e., $d_i \leq d_j$ if $i < j$. Take $k = 3$. From Corollary 1, the candidate optimal 3-partitions are limited to the following:

$$\{\{d_1, d_2, d_3\}, \{d_4, d_5, d_6\}, \{d_7, d_8, d_9\}\}$$
$$\{\{d_1, d_2, d_3, d_4\}, \{d_5, d_6, d_7, d_8, d_9\}\}$$
$$\{\{d_1, d_2, d_3, d_4, d_5\}, \{d_6, d_7, d_8, d_9\}\}.$$

Regardless of the actual values of the elements d_i, no other 3-partitions need be evaluated. In particular, 3-partitions with unconnected groups, such as

$$\{\{d_1, d_4, d_5\}, \{d_2, d_3, d_6\}, \{d_7, d_8, d_9\}\},$$

can be discarded. Three-partitions with at least one group of size ≥ 6 can also be discarded.

4 New Heuristics for Univariate Microaggregation

Practical suboptimal microaggregation methods were proposed in [3], [6], [7]. The partition mechanism is the same in all such methods: First, elements are ranked in ascending or descending order. Then, groups of consecutive k elements are combined. Inside each group, the k values taken by the variable are replaced with their average. If the total number of elements n is not a multiple of k, then the last group will contain more than k elements. As pointed out by one of the referees, a way to substantially reduce information loss with these methods is to require that the group with more than k elements be formed by elements around the modal value of the data (rather than by extreme elements as happens if the largest group is the last one).

New heuristic methods for univariate microaggregation based on the characterization of Theorem 1 and Proposition 1 will be presented in the next subsections. The new methods yield groups of variable size (at least k) and, depending on the structure of data, they may achieve lower information loss than the aforementioned proposals where the size of groups is fixed to k. Section 4.1 presents a method based on hierarchical clustering, whereas Section 4.2 discusses a genetic approach. Section 4.3 is a performance comparison of univariate methods.

4.1 Univariate Microaggregation Based on Ward Hierarchical Clustering

Since microaggregates can be viewed as a special kind of clusters, clustering methods are good candidate microaggregation methods. However, any clustering method needs to be adapted before being used for microaggregation. The reason is that the above results indicate that a way to dramatically reduce the search space for the k-partition problem implicit in microaggregation is to put constraints on both the minimal and maximal size of groups. Such constraints do not exist in the standard partition problem and must be embedded in the particular clustering method chosen.

Clustering methods can be classified as hierarchical and nonhierarchical ([2]):

- In hierarchical methods, the clustering process consists of building a hierarchical tree; by "horizontally cutting" the tree at a given level, a grouping of the initial data is obtained. A common feature of hierarchical methods is that the size of a group or cluster varies monotonically during the clustering process: Hierarchical divisive algorithms start from the root of the tree, i.e., they start with a single group, including the whole data set, which is recursively split into smaller groups thereafter; hierarchical agglomerative algorithms start from the leaves of the tree, i.e., they start with one group for each datum and begin merging those initial groups into larger groups.
- In nonhierarchical clustering, one usually distinguishes between algorithms which require that the number of resulting groups be specified in advance (fixed number of groups or FNG) and those which do not (variable number of groups or VNG). In VNG algorithms, the number of resulting groups depends on the distances between data, but *there is no constraint on the size of groups*.

When looking for a clustering method suitable for microaggregation, nonhierarchical FNG algorithms are not attractive because, in microaggregation, the number of resulting groups or microaggregates cannot be determined in advance if one attempts to minimize information loss. On the other hand, embedding constraints on the minimal and maximal group size in nonhierarchical VNG algorithms is not especially convenient because the size of groups does not vary monotonically during the clustering process; for example, the size of a group may first increase, then decrease, and then increase again. Hierarchical methods are appealing for heuristic microaggregation because they do not require the number of groups as input parameter and offer a monotonical variation of the size of groups.

Any heuristic microaggregation method should attempt to minimize the information loss L specified by (5). Since SST is fixed for a given data set, one should attempt to find a grouping that minimizes SSE. Ward's method [23] is the best-known hierarchical clustering method which attempts to minimize SSE. In addition, it has the attractive property of being *stepwise optimal* in the sense that the two groups or data elements merged at each step are chosen so that the increase in the within-groups sum of squares SSE caused by their union is minimal. See Appendix B for a description of Ward's agglomerative hierarchical clustering method.

Ward's method must be adapted to be useful for microaggregation because the original method just builds a grouping hierarchy, whereas a k-partition of the initial data set is desired.

4.1.1 Description of the Method

Ward's agglomerative hierarchical clustering method can be modified to provide a solution that belongs to the set of candidate optimal k-partitions characterized in Section 3.1. The resulting algorithm will be called k-Ward and is as follows:

Algorithm 1 (k-Ward):

1. Form a group with the first (smallest) k elements of the data set and another group with the last (largest) k elements of the data set. Intermediate elements constitute single-element groups.
2. Use Ward's method until all elements in the data set belong to a group containing k or more data elements; in the process of forming groups by Ward's method, never merge two groups which have both a size greater than or equal to k.
3. For each group in the final partition that contains $2k$ or more data elements, apply this algorithm recursively (the data set to be considered is now restricted to the particular group containing $2k$ or more elements).

The following property is proven in Appendix A.

Property 1 (Convergence). *Algorithm 1 ends after a finite number of recursion steps.*

Since we are considering one-dimensional data elements, the groups in every k-partition obtained following the above algorithm are connected. For k-Ward applied to multivariate data vectors, see Section 5.

4.1.2 Complexity and Performance of the Method

Following [2], there are two approaches to implementing Ward's algorithm: the stored matrix approach and the stored data approach.

In the stored matrix approach, the storage complexity $S_{sm}k\text{-Ward})$ of k-Ward is mainly related to the fact that Ward's clustering method requires a distance matrix containing the distance between each pair of data. Such a distance matrix is symmetrical and has zeros on its diagonal, so the storage actually needed for a data set of size n is a quadratic function of n:

$$S_{sm}(k\text{-Ward}) = (n-1)n/2. \tag{6}$$

While the stored matrix approach is very versatile, it is limited to fairly modest problems because of the size of the distance matrix. For large problems, the stored data approach is an alternative which only requires storing the original data; the values of the distance matrix are computed when needed rather than retrieved from storage. So, the storage complexity is $S_{sd}(k\text{-Ward}) = n$.

The computational complexity $C(k\text{-Ward})$ of k-Ward can be split into two components:

- The complexity $C_m(k\text{-Ward})$ of computing the distance matrix. Computation of $O(n^2)$ distance values is required, regardless of the approach used. However, the number of distance values to be computed using the stored data approach is twice that of the stored matrix approach ([2]).
- The complexity $C_c(k\text{-Ward})$ of the clustering process. This complexity is assessed below.

$C_c(k\text{-Ward})$ cannot be summarized in a formula because it depends on the structure of the particular data set being processed. However, lower and upper bounds can be found. It will be assumed that the size n of the data set is a multiple of k because this simplifies the derivation and does not affect the resulting complexity significantly. Define a *basic step* as the sequence of operations needed to merge two groups or data elements following Ward's method; $C_c(k\text{-Ward})$ can be viewed as the number of basic steps needed to complete k-Ward.

The best case for k-Ward occurs when no recursion is needed, that is, when the first k-partition obtained consists of g groups, all of which have sizes less than $2k$. Assume that the group sizes are $k_1, k_2, \cdots, k_{g-1}$ and $n - \sum_{i=1}^{g-1} k_i$. To form a group with k_i data elements, $k_i - 1$ basic steps are needed (only two groups or data elements are merged at each step). Therefore, to form the best-case k-partition, $n - g$ basic steps are needed. The larger g is, the smaller the number of basic steps needed. The largest value for g is n/k in a k-partition. Thus, the number of basic steps $C_c(k\text{-Ward})$ verifies

$$C_c(k\text{-Ward}) \geq n(1 - 1/k). \tag{7}$$

To derive the worst-case for $C_c(k\text{-Ward})$, notice that:

1. The fewer groups in a k-partition there are, the more basic steps are needed to form it. A k-partition obtained by k-Ward consists of at least two groups (by Steps 1 and 2 of Algorithm 1).
2. The more groups of $2k$ or more elements are formed, the more recursion steps of Algorithm 1 are needed.

Therefore, the worst case is when at each recursion step a two-group k-partition is obtained where one group consists of k elements and the other group contains the remaining $n - k$ elements. The first time Algorithm 1 is run takes $k - 1$ basic steps for the first group and $n - k - 1$ basic steps for the second group. If $n - k \geq 2k$, then a recursion step is needed, which will take $k - 1$ basic steps for the first group and $n - 2k - 1$ for the second group. The recursion depth will be x, where x is such that $k \leq n - kx < 2k$. If n is a multiple of k, one has $x = n/k - 1$; thus, the number of basic steps needed can be upperbounded as follows:

$$\begin{aligned} C_c(k\text{-Ward}) &\leq (n/k-1)(k-1) + \sum_{i=1}^{n/k-1} (n - ik - 1) \\ &= (n/k-1)(k-1) + (n/k-1)(n-1) - k\sum_{i=1}^{n/k-1} i \\ &= (n/k-1)(k-1) + (n/k-1)(n-1) - k\frac{(n/k-1)n/k}{2} \\ &= (n/k-1)(k-1+n-1-n/2) = (n/k-1)(n/2+k-2). \end{aligned} \tag{8}$$

The only a priori assessment that can be done regarding $C_c(k\text{-Ward})$ is that it is between the lower bound given by (7) and the upper bound given by (8); equivalently, $C_c(k\text{-Ward})$ is linear in n in the best case and is quadratic

in n in the worst case. The actual value depends on the distribution of the actual data set being microaggregated.

Thus, the overall computational complexity $C(k\text{-Ward})$ is formed by a quadratic component $C_m(k\text{-Ward})$ and a component $C_c(k\text{-Ward})$ which is linear only in the best case. This quadratic behaviour may be a problem for large n, i.e., for large data sets. A way to soften the drawback of quadratic computing time is to partition the initial data set into several connected data subsets whose size is more manageable using *k*-Ward; each data subset is then microaggregated separately.

4.2 A Genetic Approach to Microaggregation

The clustering approach presented in Section 4.1 is a viable heuristic for microaggregation but still has quadratic complexity. Genetic algorithms (GAs, [15]) appear as an alternative heuristic that can offer linear complexity (see Section 4.3 for a performance comparison). GAs combine directed and random search to locate global optima.

To use a GA, solutions must be represented as binary strings (called chromosomes). Starting from a random population of strings (initial generation), evaluation, selection, crossover, and mutation operators are applied to obtain a new and fitter generation (any optimization problem can be put as a maximization problem, in which case, the fitness of a string s is the value of the objective function in s). The *evaluation* operator computes the fitness of the current generation (average fitness over all strings or maximal fitness). The *selection* operator draws n strings with replacement from the current generation using a discrete probability distribution giving each string a weight proportional to its fitness. Given two strings, the *crossover* operator may leave them untouched with probability $1 - PCROSS$ or may cross them over (i.e., partially swap them) with probability $PCROSS$. Finally, given a string s the *mutation* operator tries to avoid confinement in a local optimum by inverting each bit of s with a (low) probability $PMUT$. This process is iterated until fitness of successive generations converges.

To illustrate the application to microaggregation, we have adapted the sequential genetic algorithm (cf. [21] or any text on genetic algorithms). Some adaptation is needed because:

- A representation of a *k*-partition as a binary string must be found.
- Crossover and mutation operators must be modified so that their output strings still represent *k*-partitions.

4.2.1 Representation of k-Partitions as Binary Strings

Assume a ranked microdata set consisting of n elements. A *k*-partition can be represented as a binary string of length n where the *i*th bit corresponds to the *i*th element in the ranked data set. In a ranked data set, from Theorem 1, all groups in the optimal *k*-partition must consist of k or more consecutive data elements. Thus, only *k*-partitions where all groups are connected need be considered. Now, if a group starts at the *i*th element, then set the *i*th bit to 1. Otherwise, set it to 0. In this way, possible solutions can be represented as *n*-bit binary strings where no two 1s are separated by less than $k-1$ zeros (no group can contain less than k data elements). On the other side, from Proposition 1, only *k*-partitions such that no two 1s are separated by more than $2k-2$ zeros need be considered in the search for the optimum. Thus, the search space of our GA can be restricted to the set of strings such that the number of zeros between any two 1s is $\geq k-1$ and $\leq 2k-2$.

4.2.2 The Crossover and Mutation Operators

Given two parent strings p_1 and p_2, the crossover operator outputs two child strings c_1 and c_2. With probability $1 - PCROSS$, one has $c_1 := p_1$ and $c_2 := p_2$. With probability $PCROSS$, crossover takes place. Crossover means that:

1. A swapping position $site$ between 0 and $n-1$ is randomly generated.
2. c_1 takes bits 0 to $site$ from p_1 and bits $site+1$ to $n-1$ from p_2.
3. c_2 takes bits 0 to $site$ from p_2 and bits $site+1$ to $n-1$ from p_1.

After crossover, the resulting child strings must be corrected to enforce that successive 1s are separated by at least $k-1$ and at most $2k-2$ zeros. This is necessary because c_1 and c_2 must represent valid minimal *k*-partitions. Corrections may be needed around the swapping position $site$. There may be two 1s around $site$ with less than $k-1$ zeros between them; in this case, the second 1 is replaced by a zero. As a result of the previous correction or simply due to crossover, a sequence of more than $2k-2$ and up to $4k-4$ successive zeros may appear after or around $site$, which may require replacing up to two zeros by 1s (the locations of the new 1s are such that they are separated by at least $k-1$ zeros).

As explained above, the goal of the mutation operator is to try to avoid confinement in a local optimum. The idea is to occasionally alter string bits with the hope of jumping into the "attraction zone" of the global optimum. Given a string s, mutation causes each bit of s to be inverted with a (low) probability $PMUT$. The string resulting after standard mutation would not necessarily represent a *k*-partition. To guarantee that a *k*-partition is obtained, a 1 is not allowed to change to zero if this is going to produce a sequence of more than $2k-2$ zeros. Similarly, a zero is not allowed to change to 1 if this would result in two 1s being separated by less than $k-1$ zeros.

4.2.3 Limitations of the Genetic Approach

A sequential genetic algorithm has a number of tunable parameters. A serious limitation of genetic algorithms is that there is no theoretical framework that allows optimal values to be found for such parameters as:

- the probabilities of crossover and mutation, $PCROSS$ and $PMUT$,
- the population size,
- the maximum number of generations,
- the convergence criterion used for stopping iteration.

Crossover is usually chosen to take place with a probability of more than 50 percent, i.e., $PCROSS > 0.5$. Mutation is chosen to occur seldom (typically, $PMUT \approx 0.05$). However, these are rules of thumb. For the number of generations and the convergence criterion used, the situation is very similar.

The convergence criterion usually consists of checking whether the fitness ratio in consecutive generations is sufficiently close to 1 (the fitness of a generation can be defined as either the average fitness of its strings or the fitness of its best-fit string).

Another weak point of genetic algorithms is that, being based on a heuristic random search, they provide solutions which are not reproducible. Running a microaggregation GA twice on the same data set will probably yield two different microaggregations.

4.2.4 Complexity and Performance of the Genetic Approach

The storage complexity $S(GA)$ of the genetic approach grows linearly with the size of n of the data set once the population size has been fixed. In fact, $POPULATION\_SIZE$ chromosomes of length n must be stored and, thus,

$$S(GA) = POPULATION\_SIZE \times n. \quad (9)$$

Low computational complexity is also a strong point of GAs. If $MAXGEN$ is the maximum number of generations allowed, then the worst-case number of basic operations is proportional to

$$MAXGEN \times POPULATION\_SIZE \times n$$

because processing a string (i.e., string generation, evaluation, crossover, and mutation) only takes a time linear in the string length. Therefore, for fixed population size and number of generations, the computational complexity $C(GA)$ of microaggregating n data elements using a sequential GA is linear in n.

An interesting property of genetic algorithms is that they are inherently parallelizable. A straightforward way to achieve parallelism is for each of a set of processors to run a sequential GA. Each processor independently creates an initial population and the subsequent generations. After some convergence criterion is met, the fittest string in all populations is taken as a solution. A number of enhancements to this basic parallel scheme exist.

4.3 Performance Comparison for Univariate Methods

The performance of the univariate microaggregation methods discussed in this paper was empirically compared as follows:

- A real data set of 834 companies in the Tarragona area was tried. For each company, 13 quantitative variables were collected: fixed assets (V1), current assets (V2), treasury (V3), uncommitted funds (V4), paid-up capital (V5), short-term debt (V6), sales (V7), labor costs (V8), depreciation (V9), operating profit (V10), financial outcome (V11), gross profit (V12), and net profit (V13). The data set corresponds to year 1995.
- Values 3 and 4 were considered for k because these are the values used in practical disclosure control. Actually, $k = 3$ is nearly always used in official statistics.
- *Individual ranking was used, that is, variables are microaggregated independently.*
- The following applies to genetic methods:
  1. Two different genetic methods were considered, depending on whether the convergence criterion for stopping iteration focuses on convergence of the average population fitness or on convergence of the maximal population fitness (fitness of the best-fit string in the population). In both cases, fitness is said to converge if the ratio between the fitnesses of two consecutive generations is ≥ 1 and $\leq 1 + 10^{-6}/n$ (remember that n is the microdata set size).
  2. The rest of the genetic parameters were set as follows: $PCROSS = 0.6$ (crossover probability), $PMUT = 0.05$ (mutation probability), $MAXGEN = 200$ (maximal number of generations), $POPULATION\_SIZE = 10$.
  3. One of the strings in the initial population used is not randomly generated, but corresponds to a *k*-partition with connected groups where all groups (except perhaps one) consist of k elements. The group around the modal value of the original data may have up to $2k - 1$ data elements. In this way, one makes sure that the final solution found by genetic random search is at least as good as the solution obtained by fixed-size univariate microaggregation [6].
  4. The best-fit string hit so far is always cached. When computation is finished (because of convergence or because the $MAXGEN$ generations have elapsed), the best-fit string is given as the final solution.

Two features were taken into account to compare methods: computing time and information loss caused by microaggregation. Section 4.3.1 reports on computing time. Section 4.3.2 reports on information loss.

4.3.1 Computing Time

The whole data set of 834 data vectors was microaggregated on a Pentium MMX at 166MHz running under the Linux operating system. Fixed-size microaggregation was performed so that the largest group is around the modal value (this is clearly the best option to reduce information loss). The following results were obtained:

- With fixed-size microaggregation, the computing time is negligible.
- With *k*-Ward, the computing time is about 25 seconds, regardless of the group size.
- With genetic methods focusing on average and maximal fitness, the computing times are 2.2 and 1.5 seconds, respectively.

It can be seen that computing time is not an issue for any method; microaggregation is usually performed off-line and even a few hundred seconds would be an acceptable time. Information loss turns out to be more interesting.

4.3.2 Information Loss

As explained at the beginning of the section, the real data set consists of 13 quantitative variables, V1 to V13, collected

TABLE 1
Comparison of Percentage Information Loss (Univariate Methods)

	$100L(k=3)$				$100L(k=4)$			
	F	k-W	A	M	F	k-W	A	M
V1	7.15	7.18	7.14	7.14	9.26	9.48	9.26	9.26
V2	.64	.56	.56	.58	1.01	.95	.81	.81
V3	.52	.78	.52	.51	1.14	1.15	1.14	1.14
V4	1.49	1.49	1.49	1.49	2.78	3.21	2.77	2.78
V5	1.69	2.02	1.69	1.69	2.11	2.11	2.11	2.11
V6	.48	.59	.48	.48	.70	.79	.70	.70
V7	1.97	1.92	1.93	1.92	3.39	3.39	3.39	3.39
V8	.42	.34	.27	.27	.59	.56	.56	.56
V9	1.29	1.29	1.29	1.29	2.80	2.90	2.79	2.79
V10	1.75	1.91	1.75	1.75	2.66	2.67	2.66	2.66
V11	2.58	2.97	2.54	2.56	3.20	3.18	3.20	3.20
V12	4.15	4.75	4.14	4.15	5.61	5.60	5.60	5.61
V13	5.01	4.96	4.98	5.01	6.68	6.68	6.65	6.68

on 834 companies. In the tables below, "F" stands for fixed-size microaggregation, with the largest group around the modal value, "k-W" for k-Ward, "A" for genetic method focusing on average fitness, and "M" for genetic method focusing on maximal fitness.

Table 1 shows the information loss L times 100 for each variable, for each method, and for $k=3$ and $k=4$. It can be observed from Table 1 that:

- For $k=3$, the A genetic method performs best for five out of 13 variables. k-W and the M genetic method perform best for three variables each. F performs best for two variables.
- For $k=4$, the A genetic method performs best for six out of 13 variables. k-W performs best for three variables. F and M genetic methods perform best for two variables.

Since the data set being considered is a multivariate one, some additional quality measures will be reported as well. More specifically, the impacts on the correlation matrix and the first principal component (justification of these quality measures can be found in [3]) are given in Table 2:

- With 13 variables, the number of unordered variable pairs is $13!/(11!2!) = 78$. Let r_{ij} be the linear correlation coefficient between variables i and j for original data; let r_{ij}^m be the correlation coefficient between the same variables once data have been microaggregated using method m. For each method m, Table 2 gives the average $\overline{\Delta r}$ and the standard deviation $s_{\Delta r}$ of the 78 discrepancies $|r_{ij}^m - r_{ij}|$. The smaller the average discrepancy and the smaller the discrepancy variability, the better a method performs.
- $\overline{\Delta W}$ is the average of percentage differences in the weight of the standardized variables on the first principal component for the original and the microaggregated data set (for each method, the average is taken over all 13 variables). The smaller $\overline{\Delta W}$, the better.
- $\%PFC$ is the percentage proportion of variability of the whole microaggregated data set explained by the first principal component. In the original data set, this percentage is 63.4; therefore, the best method is the one which results in a percentage closest to 63.4 on the microaggregated data set.

4.4 Interim Conclusions

The following conclusions can be drawn from the information loss comparison done above:

1. The performance of methods depends on the distribution of variables. No method outperforms the rest for all variables in the real data set tried. In general, we suggest routinely trying all methods for each variable and use for that variable the method yielding lowest information loss to produce the microaggregated data set to be disseminated.
2. If the data set is very large or if microaggregation is to be done online, then genetic methods offer a good trade-off between speed and information loss. But, if lack of reproducibility may cause data disclosure (as could happen in an online statistical database), then k-Ward or fixed-size microaggregation are safer.

An attractive feature of k-Ward is that it naturally extends to multivariate microaggregation (see Section 5).

5 New Heuristics for Multivariate Microaggregation

Individual ranking discussed above allows us to microaggregate the variables in a multivariate data set one at a time so that a different k-partition is obtained for each variable. Multivariate microaggregation refers to techniques which allow simultaneous microaggregation of several variables so that a single k-partition for the whole data set is obtained.

Two approaches can be taken to perform multivariate microaggregation:

Projected data: Under this approach, multivariate data are projected onto a single axis prior to microaggregation. Since projected data are univariate, they can be ranked and any of the methods discussed and Section 4 above

TABLE 2
Impact of Univariate Microaggression on Correlations and the FPC

Method	$k=3$				$k=4$			
	$\overline{\Delta r}$	$s_{\Delta r}$	$\overline{\Delta W}$	%FPC	$\overline{\Delta r}$	$s_{\Delta r}$	$\overline{\Delta W}$	%FPC
F	.0251	.0208	1.7680	64.3	.0237	.0193	1.6013	63.9
k-W	.0241	.0203	1.7164	64.3	.0254	.0206	1.6322	63.9
A	.0277	.0213	1.9676	64.7	.0251	.0202	1.6456	64.0
M	.0269	.0213	1.9481	64.5	.0249	.0203	1.6251	64.0

can be used for microaggregation. To project multivariate data vectors onto a single-axis, one can use the first principal component, the sum of z-scores of variables, a particular variable, etc. (see [3], [6], [7] for details).

Unprojected data: Heuristic methods for multivariate microaggregation are presented in this section which allow simultaneous microaggregation of several variables without resorting to one-dimensional projection.

The projected data approach has the drawback that projection adds information loss to the loss caused by microaggregation itself. The unprojected data approach has also some disadvantages, two of which are briefly commented on next:

- For unprojected data, the notion of connected group makes no sense any more. Among the theoretical results given in Section 3, only Proposition 1 remains useful.
- Adapting the genetic algorithm described in Section 4.2 for multivariate unprojected data is rather thorny: The lack of total order in multivariate spaces makes convenient representation of k-partitions as binary strings far from obvious.

Section 5.1 describes multivariate fixed-size microaggregation on unprojected data. Section 5.2 discusses multivariate microaggregation using hierarchical clustering on unprojected data. Section 5.3 contains a performance comparison of multivariate methods, both on projected and unprojected data.

5.1 Multivariate Fixed-Size Microaggregation

The method for multivariate fixed-size microaggregation described here tries to reduce information loss with respect to fixed-size microaggregation on projected data proposed in [3], [6], [7]. The method below still preserves the very low computational requirements of univariate fixed-size methods.

Algorithm 2:

1. The two most distant vectors x_r, x_s are considered and two groups are formed around them. One group contains x_r and the $k-1$ vectors closest (using the squared Euclidean distance) to x_r. The other group contains x_s and the $k-1$ vectors closest to x_s.
2. If there are at least $2k$ data vectors which do not belong to the two groups formed in Step 1, go to Step 1, taking as the new data set the previous data set minus the groups formed in the previous instance of Step 1.
3. If there are between k and $2k-1$ data vectors which do not belong to the two groups formed in Step 1, form a new group with those vectors and exit the algorithm.
4. If there are less than k data vectors which do not belong to the groups formed in Step 1, add them to the closest group formed in Step 1.

The grouping resulting from Algorithm 2 may depend on which extreme data vector one starts with, i.e., which of x_r and x_s is taken first. For example, consider the six two-dimensional data vectors in Fig. 2 and take $k=3$. The two most distant vectors are labeled 2 and 5. Starting from vector 2, the closest vector is vector 1; now, the vector closest to the group (1, 2) is vector 3. So, starting from vector 2, we get the groups (1, 2, 3) and (4, 5, 6). But, if we choose to start from the other extreme point (vector 5), the closest vector is vector 4; now, the vector closest to the group (4, 5) is vector 3. So, starting from vector 5, we get the groups (3, 4, 5) and (1, 2, 6). Anyway, the differences in the information loss resulting from starting with x_r or x_s are small. Furthermore, the larger the data set, the less likely is the kind of pathological situation depicted in Fig. 2.

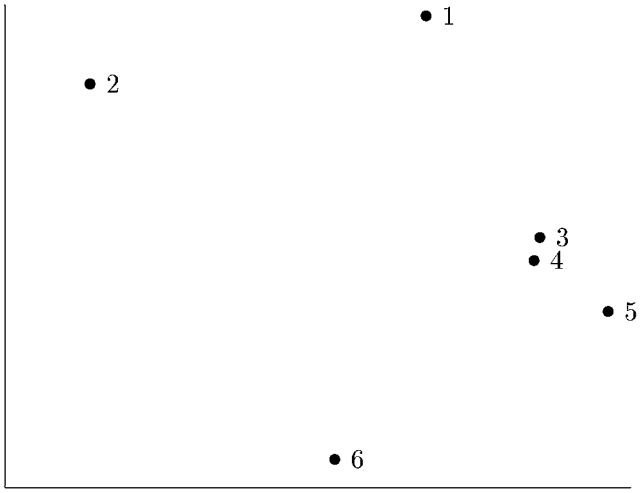

Fig. 2. Grouping with the maximum-distance criterion.

5.2 Multivariate Microaggregation Based on Ward Hierarchical Clustering

Adapting Algorithm 1 for multivariate data only requires modifying its Step 1. Basically, what is needed is a ranking criterion specifying what is meant by the "first" and "last" data vectors. Possible ranking criteria are as follows:

- *One-dimensional criteria*. Rather than using single-axis or individual ranking to perform microaggregation, as in the projected data approach, such procedures can be used only to decide which are the first and last k vectors in Step 1 of Algorithm 1.
- *Multidimensional criteria*. An example of ranking criterion which is not based on one-dimensional projection is the following maximum-distance criterion:

  1. Define as extreme data vectors the two vectors in the data set which are most distant according to the distance matrix.
  2. For each of the extreme data vectors, take the $k-1$ data vectors closest to it following the distance matrix; in this way, a group with the "first" k data vectors and another group with the "last" data vectors is obtained.

The complexity of multivariate k-Ward is the same as in the univariate case (see Section 4.1.2 above).

TABLE 3
Comparison of Percentage Information Loss
(Multivariate Methods)

Method	$100L$ ($k=3$)	$100L$ ($k=4$)
PF(PV)	30.11 - 48.48	34.14 - 57.0
PF(SZ)	28.92	32.15 or 32.08
PF(FPC)	23.87 or 23.89	30.62 or 25.99
Pk-W(PV)	31.17 - 53.93	35.28 - 58.23
Pk-W(SZ)	30.13	32.56
Pk-W(FPC)	25.35	31.71
UF	15.60	19.27
Uk-W(PV)	16.23 - 21.61	20.46 - 29.45
Uk-W(SZ)	19.16	24.31
Uk-W(FPC)	15.87	21.58
Uk-W(MD)	16.01 - 16.75	21.13 - 21.24

5.3 Performance Comparison for Multivariate Methods

We compare in this section the performance of the following methods:

PF: Multivariate fixed-size microaggregation on projected data. Projections based on a particular variable —PF(PV)—the sum of z-scores of variables—PF(SZ)— and the first principal component—PF(FPC)—are considered.

Pk-W: Multivariate k-Ward microaggregation on projected data. PV, SZ and FPC projections are considered.

UF: Multivariate fixed-size microaggregation on unprojected data (Algorithm 2).

Uk-W: Multivariate k-Ward microaggregation on unprojected data. Ranking criteria considered are PV, SZ, FPC, and also the maximum-distance criterion defined in Section 5.2, which we denote by MD.

Genetic microaggregation has not been included in the comparison because no results for unprojected data can be reported (see future research directions in Section 6).

The same data set of 834 companies described in Section 4.3 was used. Similarly to the univariate case, computing time is not an issue for multivariate methods. On a Pentium at 166MHz, the time to run multivariate fixed-size microaggregation on the whole data set is below 10 seconds, whereas multivariate k-Ward microaggregation takes less than 50 seconds. Information loss is more interesting to compare.

To compare the information loss caused by multivariate fixed-size microaggregation and k-Ward, we have considered the loss L (see (5)). It must be pointed out here that, unlike in the univariate case, the value of L depends on the units used for the variables in the microdata set. Such an undesirable property can be neutralized if all variables are standardized prior to microaggregation: If variable V_i takes a value x, then x is replaced by $(x-\bar{v}_i)/s_{v_i}$, where $\bar{v}_i$ and s_{v_i} are, respectively, the average and the standard deviation of the values taken by V_i. Results presented throughout this section have been obtained on standardized variables.

Table 3 shows the percentage values of L obtained for PF, Pk-W, UF and Uk-W.

For the FPC and SZ ranking criteria, two losses are given (unless both are very similar). The first one is obtained when data are ranked in ascending order following the criterion; the second loss is obtained when data are ranked in descending order.

A range of losses is given for the PV criterion. With this criterion, the information loss depends on the particular variable chosen for ranking. Thus, the lower limit of the range corresponds to the best variable (leading to a minimal loss) and the upper limit to the worst variable. It can be seen that the range for Uk-W is narrower than for Pk-W. In this sense, multivariate k-Ward is more *robust* on unprojected data than on projected data.

For the Uk-W(MD), a range is also given. The reason is that, each time Step 1 of the multivariate k-Ward method is run, one must decide which of two extreme data vectors is the "first" one and which is the "last" one. Thus, if Step 1 is run m times, one could in principle obtain as many as 2^m different losses. However, it can be seen that the MD criterion is fairly robust in that the difference between the worst and best losses is very small.

Table 4 is the version of Table 2 for multivariate methods (impact on correlations and first principal component). The smaller $\overline{\Delta W}$ and the closer %FPC to 63.4 percent, the better a method performs. When using the PV criterion, the figures of the variable yielding the best performance are given.

6 Conclusions and Future Research

No exact polynomial algorithms are known to date for optimal microaggregation. Methods in the literature rank data and partition them into groups of fixed-size; in the multivariate case, ranking is performed by projecting data onto a single axis. In this paper, candidate optimal solutions to the multivariate and univariate microaggregation problems have been characterized. In the univariate case, two heuristics based on hierarchical clustering and genetic algorithms have been introduced which are data-oriented in that they try to preserve natural data aggregates. In the multivariate case, fixed-size and hierarchical clustering microaggregation algorithms have been presented which do not require data to be projected onto a single dimension.

Tables 1 and 2 can hardly be compared with Tables 3 and 4. The first two tables have been discussed in Section 4.4

TABLE 4
Impact of Multivariate Microaggregation
on Correlations and the FPC

Method	$k=3$				$k=4$			
	$\overline{\Delta r}$	$s_{\Delta r}$	$\overline{\Delta W}$	%FPC	$\overline{\Delta r}$	$s_{\Delta r}$	$\overline{\Delta W}$	%FPC
PF(PV)	.20	.08	11.7	81.4	.26	.12	15.7	85.8
PF(SZ)	.24	.10	14.5	84.3	.28	.11	15.0	88.0
PF(FPC)	.20	.09	13.1	81.3	.22	.10	13.4	82.9
Pk-W(PV)	.21	.09	12.3	82.4	.27	.13	16.6	86.9
Pk-W(SZ)	.26	.10	14.1	86.0	.19	.12	15.3	88.9
Pk-W(FPC)	.22	.10	14.1	83.0	.30	.12	16.6	90.0
UF	.10	.05	7.1	71.9	.12	.06	8.1	73.7
Uk-W(PV)	.10	.06	8.0	72.6	.13	.07	9.5	75.1
Uk-W(SZ)	.13	.06	9.1	74.7	.18	.07	10.6	79.2
Uk-W(FPC)	.10	.06	8.0	72.6	.15	.07	9.9	76.7
Uk-W(MD)	.10	.05	7.9	72.1	.14	.06	9.5	75.8

and correspond to microaggregation using individual ranking on the data set, that is, dealing with each variable independently; information loss is lower than in Tables 3 and 4, where microaggregation is performed on a data vector basis (all variables simultaneously). Individual ranking usually preserves more information, but causes a higher disclosure risk than multivariate microaggregation. Indeed, with individual ranking, any intruder can rank data vectors by a particular variable and he knows that the real value of the variable for a data vector in the *i*th group is between the average of the $i-1$th group and the average of the $i+1$th group; if those two averages are very close to each other, then a very narrow interval for the real value being searched has been determined. Other safety weaknesses of individual ranking microaggregation are explored in [19].

When looking at Tables 3 and 4, it can be seen that *the multivariate methods on unprojected data proposed in this paper clearly offer lower information loss than conventional methods working on projected data.* No significant differences are evident between methods on projected data. Among methods on unprojected data, UF performs slightly better than Uk-W on average for the data set tried; however, differences being very small, they do not rule out that Uk-W might perform better for other data sets. For a given data set, we suggest trying all methods *on unprojected data* and using the method yielding lowest information loss.

Future research will be devoted to adapting genetic algorithms for multivariate microaggregation on unprojected data. To that end, a chromosome representation of *k*-partitions of unprojected data allowing efficient crossover and mutation should be devised. Using a multivariate genetic algorithm with an initial population including a *k*-partition generated by the (very fast) UF method seems especially attractive. Another topic for research is to investigate heuristics alternative to clustering methods and genetic algorithms. There are a good deal of heuristics that would seem attractive for data-oriented microaggregation (simulated annealing, tabu search, Voronoi tessellation, network models, etc.); the key point is to see whether these alternatives are able to explore the space of candidate optimal *k*-partitions in a more efficient way.

APPENDIX A

Proof (Proposition 1). The size of a group in a *k*-partition is always $\geq k$. Assume that p is an optimal *k*-partition. Assume further that the *j*th group in p has size $\geq 2k$. Then, consider the *k*-partition p' resulting from splitting the *j*th group in p into groups of size $\geq k$ and $< 2k$ and leaving the rest of the groups in p unchanged. Clearly, $SSE_{p'} \leq SSE_p$, that is, the within-groups sum of squares for p' is less than or equal to the within-groups sum of squares for p. Therefore p' is also optimal. If p' still contains a group with size $\geq 2k$, then use the same argument to obtain a p'' at least as good as p', where the group of size $\geq 2k$ has been split. Recursive application of the above argument eventually yields an optimal *k*-partition where all groups have size less than $2k$. □

Proof (Proposition 2). Let p be a minimal *k*-partition. By definition, the size of all its groups is $\geq k$. Assume that a group with size $\geq 2k$ exists. Let p' be the *k*-partition resulting from splitting such a group into groups of sizes $\geq k$ and $< 2k$ and leaving the rest of groups in p unchanged. Then, p' is finer than p, so p is not minimal unless all its groups have sizes $< 2k$.

Reciprocally, let p be a *k*-partition consisting of groups with sizes $\geq k$ and $< 2k$. If a group in p is split, then at least one of the resulting groups will have size $< k$. Therefore, any partition resulting from splitting one or more groups in p is no longer a *k*-partition. Thus, p is minimal. □

Proof (Lemma 1). In what follows, we assume that the set of n data have been ranked in ascending order. Assume that t is the total sum for the n data. For $m = k, \cdots, n-k$, consider the class P_m of *k*-partitions of the data into two groups of sizes m and $n-m$, respectively. Let X be the total sum of the group with m data, which varies over P_m. The between-groups sum-of-squares can be regarded as a parabolic function of X:

$$\begin{aligned} SSA_m(X) &= \frac{X^2}{m} + \frac{(t-X)^2}{n-m} - \frac{t^2}{n} \\ &= \frac{n}{m(n-m)}X^2 - \frac{2t}{n-m}X + \frac{t^2 m}{(n-m)n}. \end{aligned} \tag{10}$$

It is easy to see that the above formula for $SSA_m(X)$ is a special case of (3)—take $g=2$ and adapt the notation. Note that the minimum of $SSA_m(X)$ is zero and is reached at its vertex $X = tm/n$.

Now, consider a two-group *k*-partition $p \in P_m$ with at least one nonconnected group. Without loss of generality, assume that m is the cardinal of the group having a smaller total sum. Denote by x that sum, which by definition verifies $x \leq t/2$. Now, take the following partition p': One group contains the m smaller data and the other group contains the $n-m$ larger data. Let x' be the total sum of the group with the m smaller data. Clearly, $p, p' \in P_m$ and $x' \leq x$. If $x = x'$, then x is the sum of the m smaller data and, therefore, $p = p'$ and both groups in p are connected, which is against the definition of p. Therefore, in what follows we can consider $x' < x$.

1. If $x \leq tm/n$, then x is on the left of the minimum of $SSA_m(n)$ and, since $x' < x$, we have $SSA_m(x') > SSA_m(x)$. Thus, p' is better than p.
2. If $m \geq n-m$, then $tm/n \geq t/2$ and, thus,
$$x' < x \leq t/2 \leq tm/n,$$
which again yields $SSA_m(x') > SSA_m(x)$. Thus, p' is better than p.
3. If $x > tm/n$ and $m < n-m$, then consider the following partition p'': One group contains the $n-m$ smaller data and the other group contains the m larger data. Let x'' be the total sum of the group with the $n-m$ smaller data. The between-group sum-of-squares corresponding to p'' is $SSA_{n-m}(x'')$, where

$$SSA_{n-m}(X) = \frac{X^2}{n-m} + \frac{(t-X)^2}{m} - \frac{t^2}{n} = \frac{n}{(n-m)m}X^2 - \frac{2t}{m}X + \frac{t^2(n-m)}{mn}. \quad (11)$$

The minimum of $SSA_{n-m}(X)$ is zero and is reached at its vertex $X = t(n-m)/n$. In fact, $SSA_{n-m}(X)$ is nothing but $SSA_m(X)$ shifted to the right by an amount $t(n-2m)/n$ (the distance between the vertices of both parabolas). Parabolas (10) and (11) cut each other at $X = t/2$. Thus,

a. If $SSA_{n-m}(x'') > SSA_m(x)$, then p'' is better than p.
b. If $x'' < t/2$, then $SSA_{n-m}(x'') > SSA_m(x)$ because $x \leq t/2$ and, for $X'' < t/2$ and $X \leq t/2$, the parabola $SSA_{n-m}(X'')$ lies strictly above $SSA_m(X)$; also, in this subcase, p'' is better than p.
c. Otherwise,

$$tm/n < x \leq t/2 \leq x''$$

and

$$SSA_{n-m}(x'') \leq SSA_m(x).$$

Since x'' is the sum of the $n-m$ smaller data, we have $x'' \leq t(n-m)/n$, which means that x'' lies on the left of the minimum of $SSA_{n-m}(X)$. Now, x is on the right of the minimum of $SSA_m(X)$ and the only difference between both parabolas is a shift in the abscissae. Therefore, it follows from $SSA_{n-m}(x'') \leq SSA_m(x)$ that the absolute left departure of x'' from the minimum $t(n-m)/n$ must be less than or equal to the right departure of x from the minimum tm/n:

$$t(n-m)/n - x'' \leq x - tm/n.$$

The previous inequality can be simplified to $t - x \leq x''$. On the other hand, x'' is the sum of the $n-m$ smaller data; $t-x$ is the sum of a group of $n-m$ data (not necessarily the smaller data); therefore, $t - x \geq x''$. We conclude that $t - x = x''$, i.e., the group in p consisting of $n-m$ data is formed by the $n-m$ smallest data in the data set; therefore, both groups in p are connected, which is impossible by definition, so this case cannot happen.

For any $k \leq m \leq n-k$ and for any two-group k-partition $p \in P_m$ with at least one nonconnected group, we have shown a way to construct a new two-group k-partition with connected groups which is better than p. Therefore, the groups in the optimal two-group k-partition are both connected. □

Proof (Lemma 2). Let SSA_p be the between-groups sum-of-squares for the optimal partition consisting of groups $1, 2, \cdots, g$. Let SSA_{p_g} be the between-groups sum-of-squares for the partition p_g resulting from suppression of the gth group (without loss of generality, we can assume that the suppressed group is the last one). From (3), we have:

$$SSA_p = SSA_{p_g} + \frac{(t-t_g)^2}{n-n_g} + \frac{t_g^2}{n_g} - \frac{t^2}{n}.$$

Assume that p_g is not optimal over the data comprised in groups $1, 2, \cdots, g-1$; then, a p'_g exists over those data such that $SSA_{p'_g} > SSA_{p_g}$. Therefore,

$$SSA_p < SSA_{p'_g} + \frac{(t-t_g)^2}{n-n_g} + \frac{t_g^2}{n_g} - \frac{t^2}{n},$$

which means that the partition formed by p'_g and the gth group is better than p. But, p is optimal, so a p'_g better than p_g cannot exist. Thus, p_g is also optimal. □

Proof (Theorem 1). Let p be an optimal k-partition for the whole set of n data. Consider the n data elements ranked in ascending order. Let $\{G_1, \cdots, G_g\}$ be the groups in p. If all groups are connected, then the result follows. Otherwise, assume that there are some elements of G_j between the elements G_i, where $j \neq i$. In this case, consider the k-partition $p_{i,j} = \{G_i, G_j\}$ over its corresponding data subset. Given that p is optimal, recursive use of Lemma 2 yields that $p_{i,j}$ is optimal over its corresponding data subset. Now, by Lemma 1, if $p_{i,j}$ is optimal, then both G_i and G_j must be connected, which is against the assumption that there are elements of G_j between elements of G_i. Therefore, all groups in p must be connected. □

Proof (Property 1). By Step 1 of the above algorithm, each new recursion step starts splitting the initial data group into at least two groups; the rule in Step 2 ensures that the group formed by the smallest elements and the group formed by the largest elements are never merged thereafter (because of their sizes). In this way, at the end of a recursion step, the final k-partition consists of at least two groups and is therefore finer than the initial k-partition (consisting of a single group). If there is a group of size $\geq 2k$, then the algorithm is recursively applied to it and strictly smaller groups will be obtained (according to the previous argument). Thus, after a finite number of recursion steps, a k-partition of the initial data set will be obtained such that the maximal group size is less than $2k$. □

APPENDIX B
THE WARD METHOD

An agglomerative hierarchical clustering method produces a series of partitions of the data $p_n, p_{n-1}, \cdots, p_1$. The first, p_n, consists of n single-element groups or clusters; the last, p_1, consists of a single group containing all n elements. The basic operation of this family of methods is:

Algorithm 3.

1. Consider groups $G_1, G_2, \cdots, G_n$, each containing a single element.
2. Find the nearest pair of distinct groups, G_i and G_j, merge G_i and G_j, delete G_j, and decrement the number of groups by one.

3. If the number of groups is one, then stop, else return to Step 2.

At each particular stage, the methods merge elements or groups which are closest (or most similar). Differences between methods arise because of the different ways of defining distance (or similarity) between an element and a group containing several elements or between two groups of elements.

Ward [23] proposed a clustering procedure seeking to perform the partitions $p_n, p_{n-1}, \cdots, p_1$ in a manner that minimizes the loss associated with each grouping and to quantify that loss in a form that is readily interpretable. At each iteration in the algorithm, the union of every possible pair of groups is considered and the two groups whose fusion results in the minimum increase in "information loss" are combined. Information loss is defined by Ward in terms of within-groups sum-of-squares SSE.

In the Ward method, the distance between two groups is defined as the increase in the within-groups sum of squares SSE that would happen if both groups were merged. To illustrate, let us consider the univariate case. Initially, when all groups consist of a single element, $SSE = 0$. The distance $d(x, y)$ between two univariate data elements x and y is exactly

$$d(x,y) = \left(x - \frac{x+y}{2}\right)^2 + \left(y - \frac{x+y}{2}\right)^2 = (x-y)^2/2.$$

Similarly, the distance between two groups G_i and G_j with n_i and n_j elements, respectively, can be computed as

$$d(G_i, G_j) = \frac{n_i n_j}{n_i + n_j}(\bar{x}_i - \bar{x}_j)^2,$$

where $\bar{x}_i$ is the average of the elements in G_i and $\bar{x}_j$ is the average of the elements in G_j. At each step of the method, the groups chosen for merging are those with minimal distance between them. When two groups are merged, the distances from the resulting group to the remaining groups must be recomputed.

ACKNOWLEDGMENTS

This work was partly supported by the Spanish CICYT under grant nos. TIC95-0903-C02-02 and TEL98-0699-C02-02 and by the US Bureau of the Census under contract no. OBLIG-2000-29158-0-0. Also, the three anonymous referees have greatly contributed to improve the clarity, the organization and the contents of this paper. In particular, the authors acknowledge the following suggestions: a way to enhance fixed-size microaggregation by reducing information loss; the possibility of considering different weights for each variable to compute information loss; the need to include significant references.

REFERENCES

[1] N.R. Adam and J.C. Wortmann, "Security-Control Methods for Statistical Databases: A Comparative Study," *ACM Computing Surveys,* vol. 21, pp. 515-556, 1989.
[2] M.R. Anderberg, *Cluster Analysis for Applications.* New York: Academic Press, 1973.
[3] N. Anwar, "Micro-Aggregation—The Small Aggregates Method," internal report, Luxembourg: Eurostat, 1993.
[4] J.G. Bethlehem, W.J. Keller, and J. Pannekoek, "Disclosure Control of Microdata," *J. Am. Statistical Assoc.,* vol. 85, pp. 38-45, 1990.
[5] P. Brucker, "On the Complexity of Clustering Problems," *Lecture Notes in Economics and Math. Systems,* vol. 157, pp. 45-54, Berlin: Springer-Verlag, 1978.
[6] D. Defays and P. Nanopoulos, "Panels of Enterprises and Confidentiality: The Small Aggregates Method," *Proc. '92 Symp. Design and Analysis of Longitudinal Surveys,* pp. 195-204, 1993.
[7] D. Defays and N. Anwar, "Micro-Aggregation: A Generic Method," *Proc. Second Int'l Symp. Statistical Confidentiality,* pp. 69-78, 1995.
[8] D.E. Denning, *Cryptography and Data Security.* Reading, Mass.: Addison-Wesley, 1982.
[9] G.T. Duncan and D. Lambert, "Disclosure-Lmited Data Dissemination," *J. Am. Statistical Assoc.,* vol. 81, pp. 10-28, 1986.
[10] A.W.F. Edwards and L.L. Cavalli-Sforza, "A Method for Cluster Analysis," *Biometrics,* vol. 21, pp. 362-375, 1965.
[11] W.D. Fisher, "On Grouping for Maximum Homogeneity," *J. Am. Statistical Assoc.,* vol. 53, pp. 789-798, 1958.
[12] A.D. Gordon and J.T. Henderson, "An Algorithm for Euclidean Sum of Squares Cassification," *Biometrics,* vol. 33, pp. 355-362, 1977.
[13] P. Hansen, B. Jaumard, and N. Mladenovic, "Minimum Sum of Squares Clustering in a Low Dimensional Space," *J. Classification,* vol. 15, pp. 37-55, 1998.
[14] J.A. Hartigan, *Clustering Algorithms.* New York: Wiley, 1975.
[15] J.H. Holland, "Genetic Algorithms and the Optimal Allocation of Trials," *SIAM J. Computing,* vol. 2, no. 2, pp. 88-105, 1973.
[16] R.C. Jancey, "Multidimensional Group Analysis," *Australian J. Botany,* vol. 14, pp. 127-130, 1966.
[17] J.B. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," *Proc. Fifth Berkeley Symp. Math. Statistics and Probability,* vol. 1, pp. 281-297, 1967.
[18] J.M. Mateo-Sanz and J. Domingo-Ferrer, "A Method for Data-Oriented Multivariate Microaggregation," *Proc. Statistical Data Protection '98,* pp. 89-99, 1999.
[19] J. Mateo-Sanz and J. Domingo-Ferrer, "Microaggregation with Individual Ranking: Cautionary Remarks on Security," manuscript.
[20] G. Paass, "Disclosure Risk and Disclosure Avoidance for Microdata," *J. Business and Economic Studies,* vol. 6, pp. 487-500, 1988.
[21] J.L. Ribeiro Filho and P.C. Treleaven, "Genetic-Algorithm Programming Environments," *Computer,* vol. 27, pp. 28-43, 1994.
[22] C.J. Skinner, "On Identification Disclosure and Prediction Disclosure for Microdata," *Statistica Neerlandica,* vol. 46, pp. 21-32, 1992.
[23] J.H. Ward, "Hierarchical Grouping to Optimize an Objective Function," *J. Am. Statistical Assoc.,* vol. 58, pp. 236-244, 1963.
[24] L. Willenborg and T. de Waal, *Statistical Disclosure Control in Practice.* New York: Springer-Verlag, 1996.

Josep Domingo-Ferrer (M'89) received the MSc and PhD degrees in computer science with honors from the Universitat Autònoma de Barcelona in 1988 and 1991, respectively. In 1995, he also earned an MSc degree in mathematics from U.N.E.D., Madrid. He is currently an associate professor in the Department of Computer Engineering and Mathematics of the Universitat Rovira i Virgili, Tarragona, Catalonia, Spain. His fields of expertise are statistical disclosure control, cryptography, and data security. He has authored one patent and more than 60 publications. He has chaired the program committees of Statistical Data Protection '98 (sponsored by Eurostat) and IFIP CARDIS 2000. He is currently a guest editor of *Computer Networks* and serves as chairman of the IEEE Information Theory Society Spanish Chapter. He is a member of the IEEE.

Josep Maria Mateo-Sanz received the MSc and PhD degrees in mathematics from the Universitat de Barcelona in 1992 and 1998, respectively. Dr. Mateo-Sanz is currently a tenure-track associate professor of statistics in the Department of Chemical Engineering of the Universitat Rovira i Virgili, Tarragona, Catalonia, Spain. His interests are in statistical disclosure control and applied multivariate statistics.