An Optimization-Based Decomposition Heuristic for the Microaggregation Problem

Jordi Castro[1](✉), Claudio Gentile[2], and Enric Spagnolo-Arrizabalaga[1]

[1] Department of Statistics and Operations Research, Universitat Politècnica de Catalunya, Jordi Girona 1–3, 08034 Barcelona, Catalonia
jordi.castro@upc.edu, gentile@iasi.cnr.it

[2] Istituto di Analisi dei Sistemi ed Informatica "A. Ruberti", Consiglio Nazionale delle Ricerche, Rome, Italy

Abstract. Given a set of points, the microaggregation problem aims to find a clustering with a minimum *sum of squared errors* (SSE), where the cardinality of each cluster is greater than or equal to k. Points in the cluster are replaced by the cluster centroid, thus satisfying k-anonymity. Microaggregation is considered one of the most effective techniques for numerical microdata protection. Traditionally, non-optimal solutions to the microaggregation problem are obtained by heuristic approaches. Recently, the authors of this paper presented a mixed integer linear optimization (MILO) approach based on column generation for computing tight solutions and lower bounds to the microaggregation problem. However, MILO can be computationally expensive for large datasets. In this work we present a new heuristic that combines three blocks: (1) a decomposition of the dataset into subsets, (2) the MILO column generation algorithm applied to each dataset in order to obtain a valid microaggregation, and (3) a local search improvement algorithm to get the final clustering. Preliminary computational results show that this approach was able to provide (and even improve upon) some of the best solutions (i.e., of smallest SSE) reported in the literature for the Tarragona and Census datasets, and $k \in \{3, 5, 10\}$.

Keywords: Statistical disclosure control · Microdata · Microaggregation problem · Mixed integer linear optimization · Column generation · Local search · Heuristics

1 Introduction

A microdata file of p individuals (people, companies, etc.) and d variables (or attributes) is, in practice, a matrix $A \in \mathbb{R}^{p \times d}$ whose element a_{ij} provides the value of attribute j for individual i, and whose row a_i gives the d attributes for

Supported by grant MCIU/AEI/FEDER RTI2018-097580-B-I00.

individual i. Formally, a microdata file is a mapping $M : S \subseteq P \rightarrow V_1 \times \ldots \times V_t$, where P is a population, S is a sample of the population and V_i is the domain of the attribute $i \in \{1, \ldots, d\}$.

Microdata files must be protected before being released; otherwise, confidential individual information would be jeopardized. Microaggregation [5,6] is a statistical disclosure control technique, mainly for numeric variables, which is related with *k-anonymity* [20].

The goal of microaggregation is to modify the values of the variables such that the released microdata satisfies k-anonymity. Therefore, it first partitions the individuals (or points in $\mathbb{R}^d$) into subsets of size at least k, called *clusters*, and it then replaces each point in the cluster with the centroid of the cluster in order to minimize the loss of information, called *spread*. In practical cases, the value of k is relatively small (e.g., $3 \leq k \leq 10$, see [6]). A widely used measure for evaluating the spread is the *sum of squared errors* (SSE) [6]:

$$SSE = \sum_{i=1}^{q} \sum_{j=1}^{n_i} (a_{i_j} - \overline{a}_i)^T (a_{i_j} - \overline{a}_i), \tag{1}$$

where q denotes the number of clusters, n_i the size of cluster $\mathcal{C}_i = \{a_{i_j}, j = 1, \ldots, n_i\}$, and $\overline{a}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} a_{i_j}$ its centroid, for $i = 1, \ldots, q$. An equivalent measure that is also widely used in the literature is the *information loss* (IL), which is defined as

$$IL = \frac{SSE}{SST} \cdot 100, \tag{2}$$

where SST is the total sum of squared errors for all the points, that is:

$$SST = \sum_{i=1}^{p} (a_i - \bar{a})^\top (a_i - \bar{a}) \quad \text{where } \bar{a} = \frac{\sum_{i=1}^{p} a_i}{p}. \tag{3}$$

IL always takes values within the range $[0, 100]$; the smaller the IL, the better the clustering. From now on, we will denote as *feasible clustering* a partition into clusters of size at least k.

Finding the partition that minimizes IL (or SSE) and satisfies the cardinality requirement $n_i \geq k$ for $i = 1, \ldots, q$ is a difficult combinatorial optimization problem when $d > 1$ (multivariate data), which is known to be NP-hard [15]. For univariate data (that is, $d = 1$)—which in practice are the exception—microaggregation can be solved in polynomial time using the algorithm of [11], which is based on the shortest path problem.

Microaggregation differs from other related clustering problems, such as k-medians or k-means [10], specifically in that it imposes a lower bound k on the cardinality of each cluster, but no fixed number of clusters. On the other hand, k in k-medians and k-means fixes the number of clusters, while imposing no constraint on the cardinality of each cluster.

There exist various papers on heuristic algorithms for feasible solutions to multivariate microaggregation with reasonable IL. Heuristics like *minimum distance to average* (MDAV) [6,8] and *variable minimum distance to average*

(VMDAV) [19] sequentially build groups of fixed (MDAV) or variable (VMDAV) size based on considering the distances of the points to their centroid. Other approaches first order the multivariate points and apply the polynomial time algorithm of [11] to the ordered set of points, such as in [7], which used several fast ordering algorithms based on paths in the graph that is associated with the set of points, whereas [14] used (slower) Hamiltonian paths (which involve the solution of a traveling salesman problem). The heuristic of [16] also sequentially builds a set of clusters attempting to locally minimize IL. Other approaches, such as those of [2,13], are based on refining the solutions previously provided by another heuristic.

To our knowledge, the only two papers in the literature to apply optimization techniques to microaggregation and formulate it as a combinatorial optimization problem are [1] and [4]. Both of them apply a column generation algorithm inspired by the work in [9]. Those optimization approaches solve the linear relaxation of the integer microaggregation problem, thus computing a (usually tight) lower bound for the problem. They also provide a (usually very good) upper bound solution with an IL that is smaller than the ones reported by other heuristics. Note that having a lower bound of the optimal solution is instrumental in order to know how good are the (upper bound) solutions computed by heuristics, even to perform fair comparisons between them. For instance, the heuristic introduced in [17] reported IL values below the certified lower bound—thus, not possible—, which clearly indicates that the values of the datasets used in that paper were different than those in the rest of the literature (likely due to some sort of normalization of attributes).

The downside of those optimization based techniques is that, when the dataset is large, the column generation may involve a large number of iterations, thus making it computationally very expensive. The main difference between the approaches in [1] and [4] is that the pricing problem of the former involves a nonlinear integer problem while the latter requires a simpler linear integer problem. In practice this means that the pricing subproblem in [1] can be tackled only by means of complete enumeration and only for small values of k, while [4] theoretically offers more flexibility and can deal with larger values of k.

Since optimization-based methods can be inefficient for large datasets but can provide high quality solutions in reasonable time for microdata with a small number of points, this work suggests a new approach consisting of first partitioning the set of points, and then applying an optimization approach to each smaller subset. The initial partitioning of the dataset is done according to a feasible clustering previously computed with the MDAV/VMDAV heuristics. Additionally, a local search improvement heuristic is also applied twice: first, to the solution provided by the MDAV/VMDAV heuristics prior to the partitioning; and second, to the final solution provided by the optimization approach.

This short paper is organized as follows. Section 2 outlines the optimization-based decomposition heuristic. Sections 3 and 4 outline the two main building blocks of the heuristic: the local search improvement algorithm and the mixed integer linear optimization method based on column generation in [4]. Section 5 shows the preliminary results from this approach with the standard Tarragona and Census datasets used in the literature.

2 The Decomposition Heuristic

The decomposition heuristic comprises the following steps:

Input: Microdata matrix $A^0 \in \mathbb{R}^{p \times d}$, minimum cluster cardinality k, upper bound of the number of subsets s in which the microdata will be decomposed.

1. Standardize attributes/columns of A^0, obtaining matrix $A \in \mathbb{R}^{p \times d}$. Compute the squared Euclidean distance matrix $D \in \mathbb{R}^{p \times p}$, where $D_{ij} = (a_i - a_j)^\top (a_i - a_j)$, which is to be used in the remaining steps.
2. Apply MDAV and VMDAV microaggregation heuristics using D. Let $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_q\}$ be the best of the two feasible clusterings provided by MDAV and VMDAV (that is, the one with smallest IL). Here, q represents the number of clusters, and $\mathcal{C}_i$ the set of points in cluster i.
3. Apply the local search improvement algorithm (described in Sect. 3) to $\mathcal{C}$, obtaining the updated clustering $\mathcal{C}' = \{\mathcal{C}'_1, \ldots, \mathcal{C}'_q\}$. The updated clustering $\mathcal{C}'$ has the same number of clusters q as $\mathcal{C}$, but the points in subsets $\mathcal{C}'_i$ and $\mathcal{C}_i$ can be different.
4. Partition the microdata and distance matrices A and D in $s' \leq s$ subsets $\mathcal{S}_i, i = 1, \ldots, s'$, according to the clustering $\mathcal{C}'$. For this purpose we compute $\bar{p} = round(p/s)$, the minimum number of points in each subset of the partition, and build each subset $\mathcal{S}_i$ by sequentially adding points of clusters $\mathcal{C}'_j, j = 1, \ldots, q$ until the cardinality $\bar{p}$ is reached.
5. Apply the mixed integer linear optimization method based on column generation in [4] to each microaggregation subproblem defined by $A_{\mathcal{S}_i}$ and $D_{\mathcal{S}_i}$, $i = 1, \ldots, s'$. Obtain feasible clustering $\mathcal{O}_i$ for points in $\mathcal{S}_i$, $i = 1, \ldots, s'$.
6. Perform the union of clusterings $\mathcal{O} = \mathcal{O}_1 \cup \cdots \cup \mathcal{O}_{s'}$. $\mathcal{O}$ is a feasible clustering for the original microdata A.
7. Finally, once again apply the local search improvement algorithm from Sect. 3 to $\mathcal{O}$ in order to obtain the final microaggregation $\mathcal{O}'$.

Return: Clustering $\mathcal{O}'$.

Step 3 of the algorithm can be skipped, thus obtaining the partition in Step 4 with the clustering $\mathcal{C}$ from Step 2. However, we have observed that better results are generally obtained if the local search improvement heuristic is applied in both Steps 3 and 7, not only in Step 7. Indeed, if efficiency is a concern, it is possible to stop the whole procedure after Step 3, which thus returns cluster $\mathcal{C}'$ as a solution and, in general, significantly outperforms the solution obtained in Step 2. In this way, it is possible to avoid Step 5, which is usually computationally expensive.

Note also that the clustering $\mathcal{C}'$ obtained in Step 3 is used only in Step 4 to decompose the microdata into subsets, but not as a starting solution for

```
Algorithm Two-swapping heuristic($\mathcal{C}$: initial feasible clustering))
    bestSSE= currentSSE= $SSE(\mathcal{C})$
    Repeat
        for $i = 1$ to $p$
            for $j = i + 1$ to $p$
                if swapping $i$ and $j$ improves bestSSE then
                    Update best swapping: $i' = i$, $j' = j$, update bestSSE
                end_if
            end_for
        end_for
        improvement= (bestSSE < currentSSE)
        if improvement then
            Update $\mathcal{C}$ by swapping points $i'$ and $j'$
            currentSSE = bestSSE
        end_if
    while improvement
    Return: $\mathcal{C}$ updated with swaps
End_algorithm
```

Fig. 1. Two-swapping local search improvement heuristic

the optimization procedure in Step 5. Therefore, the clustering computed in Steps 5–6 by the optimization procedure might have a larger SSE than $\mathcal{C}'$. On the other hand, by not starting the optimization procedure in Step 5 with the solution $\mathcal{C}'$ we have some chances to obtain a different and possibly better local solution. In the current implementation, $\mathcal{C}'$ is not used as a starting solution for the optimization algorithm.

The larger the value of s, the faster the algorithm will be, since the minimum number of points $\bar{p} = round(p/s)$ in each subset $\mathcal{S}_i, i = 1, \ldots, s'$ will be smaller, and therefore the optimization algorithm of [4] will be more efficient. However, the final IL (SSE) of the final clustering $\mathcal{O}'$ also increases with s. Therefore parameter s can be used as a trade-off between efficiency and solution quality.

In the next two sections, we outline the local search improvement heuristic used in Steps 3 and 7, as well as the mixed integer linear optimization method in Step 5.

3 The Local Search Improvement Heuristic

Given a feasible clustering for the microaggregation problem, a local search algorithm tries to improve it by finding alternative solutions in a local neighborhood of the current solution. The local search considered in this work is a two-swapping procedure; in addition to its simplicity, it has proven to be very effective in practice. Briefly, the two-swapping heuristic performs a series of iterations, and at each iteration it finds the pair of points (i, j) located in different clusters that would most reduce the overall SSE if they were swapped. This operation is repeated until no improvement in SSE is detected. The cost per iteration of the heuristic is $O(p^2/2)$. Similar approaches have been used in other clustering techniques, such as in the partitioning around medoids algorithm for k-medoids [12]. The two-swapping algorithm implemented is shown in Fig. 1.

4 The Mixed Integer Linear Optimization Algorithm Based on Column Generation

In this section we quickly outline the optimization method presented in [4]. Additional details can be found in that reference.

The formulation of microaggregation as an optimization problem in [4] is based on the following property of the SSE_h of cluster $\mathcal{C}_h = \{a_{h_i}, i = 1, \dots, n_h\}$ (see [4, Prop. 3] for a proof):

$$\begin{aligned} SSE_h &= \sum_{i=1}^{n_h} (a_{h_i} - \overline{a_h})^\top (a_{h_i} - \overline{a_h}) \\ &= \frac{1}{2n_h} \sum_{i=1}^{n_h} \sum_{j=1}^{n_h} (a_{h_i} - a_{h_j})^\top (a_{h_i} - a_{h_j}) = \frac{1}{2n_h} \sum_{i=1}^{n_h} \sum_{j=1}^{n_h} D_{h_i h_j}. \end{aligned} \tag{4}$$

That is, for computing SSE_h, we do not need the centroid of the cluster, but only the distances between the points in the cluster.

From (4), defining binary variables x_{ij}, $i, j = 1, \dots, p$ (which are 1 if points i and j are in the same cluster, 0 otherwise), then the microaggregation problem can be formulated as:

$$\min \quad SSE \triangleq \frac{1}{2} \sum_{i=1}^{p} \frac{\sum_{j=1, j\neq i}^{p} D_{ij} x_{ij}}{\sum_{j=1, j\neq i}^{p} x_{ij} + 1} \tag{5a}$$

$$\text{s. to} \quad x_{ir} + x_{jr} - x_{ij} \leq 1 \quad i, j, r = 1, \dots, p, \quad i \neq j, r \neq j, i \neq r \tag{5b}$$

$$\sum_{j=1, j\neq i}^{p} x_{ij} \geq k - 1 \quad i = 1, \dots, p \tag{5c}$$

$$x_{ij} = x_{ji}, x_{ij} \in \{0, 1\}, i, j = 1, \dots, p. \tag{5d}$$

Constraints (5b) are triangular inequalities, that is, if points i and r, and r and j are in the same cluster, then points i and j are also in the same cluster. Constraints (5c) guarantee that the cardinality of the cluster is at least k. The denominator in the objective function (5a) is the cardinality of the cluster that contains point i. Unfortunately, (5) is a difficult nonlinear and nonconvex integer optimization problem (see [4] for details).

A more practical alternative is to consider a formulation inspired by the clique partitioning problem with minimum clique size of [9]. Defining as $\mathcal{C}^* = \{\mathcal{C} \subseteq$

$\{1,\ldots,p\} : k \leq |\mathcal{C}| \leq 2k-1\}$ the set of feasible clusters, the microaggregation problem can be formulated as:

$$\begin{aligned} \min \; & \sum_{\mathcal{C}\in\mathcal{C}^*} w_{\mathcal{C}} x_{\mathcal{C}} \\ \text{s. to} \; & \sum_{\mathcal{C}\in\mathcal{C}^*: i\in\mathcal{C}} x_{\mathcal{C}} = 1 \; i \in \{1,\ldots,p\} \\ & x_{\mathcal{C}} \in \{0,1\} \qquad \mathcal{C}\in\mathcal{C}^*, \end{aligned} \tag{6}$$

where $x_{\mathcal{C}} = 1$ means that feasible cluster $\mathcal{C}$ appears in the microaggregation solution provided, and the constraints guarantee that all the points are covered by some feasible cluster.

From (4), the cost $w_{\mathcal{C}}$ of cluster $\mathcal{C}$ in the objective function of (6) is

$$w_C = \frac{1}{2\,|C|} \sum_{i\in\mathcal{C}} \sum_{j\in\mathcal{C}} D_{ij}. \tag{7}$$

The number of feasible clusters in $\mathcal{C}^*$—that is, the number of variables in the optimization problem (6)—is $\sum_{j=k}^{2k-1} \binom{p}{j}$, which can be huge. For instance, for $p = 1000$ and $k = 3$ we have $|\mathcal{C}^*| = 8.29 \cdot 10^{12}$. However, the linear relaxation of (6) can be solved using a column generation technique, where the master problem is defined as (6) but it considers only a subset $\bar{\mathcal{C}} \subseteq \mathcal{C}^*$ of the variables/clusters. The set $\bar{\mathcal{C}}$ is updated at each iteration of the column generation algorithm with new clusters, which are computed by a pricing subproblem. The pricing subproblem either detects that the current set $\bar{\mathcal{C}}$ contains the optimal set of columns/clusters or, otherwise, it generates new candidate clusters with negative reduced costs. For small datasets and values of k, the pricing subproblem can be solved by complete enumeration; otherwise, an integer optimization model must be solved. The master problem requires the solution of a linear optimization problem. Both the linear and integer optimization problems were solved with CPLEX in this work. The solution of the linear relaxation of (6) provides a lower bound to the microaggregation problem (usually a tight lower bound). In addition, solving the master problem as an integer problem allows us to obtain a feasible solution to the microaggregation problem (usually of high quality). A thorough description of this procedure, and of the properties of the pricing subproblem, can be found in [4] and [18].

5 Computational Results

The algorithm in Sect. 2 and the local search heuristic in Sect. 3 have been implemented in C++. We used the code of [4] (also implemented in C++) for the solution of the mixed integer linear optimization approach based on column generation, as described in Sect. 4. A time limit of 3600 s was set for the solution of each subproblem with the column generation algorithm in Step 5 of the heuristic in Sect. 2. We tested the decomposition algorithm with the datasets Tarragona and Census, which are standard in the literature [3]. The results for Tarragona and Census are shown, respectively, in Tables 1–2 and 3–4. These tables show, for each instance and value $k \in \{3, 5, 10\}$, the IL and CPU time for the main steps of the decomposition heuristic in Sect. 2. We also tried the different values $s \in \{40, 20, 10, 5, 2\}$ for partitioning the dataset in Step 5. For Step 2, the tables also show which of the MDAV or VMDAV algorithms reported the best solution. The difference between Tables 1 and 2 (also between Tables 3 and 4) is that the former show results with the Step 3, while in the latter this step was skipped. We remind the reader that the decomposition algorithm could be stopped after Step 3 with a feasible and generally good solution. However, the best IL values for each k, which are marked in boldface in the tables, are obtained after Step 7, although this means going through the usually more expensive Step 5.

Table 1. Results for the Tarragona dataset considering Step 3 of the algorithm. The best IL for each k is marked in boldface.

Instance	k	Step 2			Step 3		Step 5			Step 7	
		Alg	IL	CPU	IL	CPU	s	IL	CPU	IL	CPU
Tarragona	3	VMDAV	15.85	0.6	15.00	1.9	40	14.96	0.2	14.85	0.1
							20	14.83	0.4	14.81	0.1
							10	14.68	0.9	14.65	0.3
							5	14.57	2.8	14.53	0.4
							2	14.51	4.2	**14.50**	0.2
Tarragona	5	MDAV	22.46	0.5	20.74	5.2	40	20.74	1.0	20.73	0.5
							20	20.74	7.9	20.73	0.3
							10	20.47	103.8	20.40	1.5
							5	20.32	1119.6	**20.25**	1.6
							2	21.06	7207.8	20.46	4.0
Tarragona	10	MDAV	33.19	0.3	30.77	25.6	40	30.77	12119.7	30.77	0.1
							20	33.03	61600.3	30.87	9.0
							10	31.80	88899.1	**30.56**	13.4
							5	33.32	9.8	30.79	17.4
							2	33.20	22.8	30.80	17.2

From Tables 1, 2, 3 and 4 we conclude that:

- In general, the smaller the k and larger the s, the faster the decomposition heuristic. In a few cases, however, smaller values of s also meant smaller CPU times; for instance, this is observed for Census, $k = 10$, and values $s = 20$ and $s = 10$. The explanation is that the maximum time limit was reached in some pricing subproblems in those runs, and therefore the CPU time increased with s.
- In general, smaller ILs are obtained for smaller s values, as expected.

Table 2. Results for the Tarragona dataset without Step 3 of the algorithm. The best IL for each k is marked in boldface.

Instance	k	Step 2			Step 5			Step 7	
		Alg	IL	CPU	s	IL	CPU	IL	CPU
Tarragona	3	VMDAV	15.85	0.6	40	15.15	0.2	14.95	1.7
					20	14.86	0.3	14.73	1.3
					10	14.75	1.0	14.66	1.2
					5	14.58	1.9	14.54	0.7
					2	14.51	4.6	**14.50**	0.4
Tarragona	5	MDAV	22.46	0.5	40	21.81	0.9	21.18	4.7
					20	21.14	7.9	20.59	4.7
					10	20.62	86.2	20.36	3.8
					5	20.37	894.5	**20.29**	2.1
					2	21.01	7208.1	20.77	6.0
Tarragona	10	MDAV	33.19	0.3	40	32.05	12597.2	30.82	19.9
					20	33.18	60243.5	30.81	20.8
					10	32.23	230644.6	**30.55**	20.9
					5	33.20	13.0	30.84	20.9
					2	33.21	21.2	30.83	20.6

Table 3. Results for the Census dataset considering Step 3 of the algorithm. The best IL for each k is marked in boldface.

Instance	k	Step 2			Step 3		Step 5			Step 7	
		Alg	IL	CPU	IL	CPU	s	IL	CPU	IL	CPU
Census	3	VMDAV	5.66	1.2	5.25	3.3	40	5.21	0.1	5.20	0.2
							20	5.20	0.2	5.19	0.1
							10	5.21	0.6	5.18	0.3
							5	5.12	3.2	5.07	0.8
							2	4.85	5.2	**4.79**	0.4
Census	5	VMDAV	8.98	1.1	8.12	8.7	40	8.12	2.4	8.12	0.2
							20	8.12	22.5	8.11	0.2
							10	8.14	247.0	8.09	0.8
							5	7.96	5334.6	**7.84**	2.0
							2	9.36	7209.3	8.19	8.5
Census	10	VMDAV	14.04	0.8	12.36	38.4	40	12.36	8452.7	12.36	0.4
							20	12.96	77221.1	**12.32**	7.0
							10	12.84	37037.7	12.40	14.5
							5	13.49	7310.4	12.63	28.8
							2	14.45	26.3	12.46	37.1

Table 4. Results for the Census dataset without Step 3 of the algorithm. The best IL for each k is marked in boldface.

Instance	k	Step 2			Step 5			Step 7	
		Alg	IL	CPU	s	IL	CPU	IL	CPU
Census	3	VMDAV	5.66	1.2	40	5.56	0.1	5.19	3.2
					20	5.51	0.3	5.19	2.8
					10	5.44	0.9	5.19	2.1
					5	5.26	2.3	5.10	1.4
					2	4.87	4.9	**4.81**	0.7
Census	5	VMDAV	8.98	1.1	40	8.94	2.2	8.14	8.3
					20	8.84	22.3	8.10	8.2
					10	8.63	218.6	8.10	5.9
					5	8.22	4050.7	**7.90**	3.9
					2	9.13	7210.2	8.19	8.1
Census	10	VMDAV	14.04	0.8	40	13.88	7647.9	12.47	31.8
					20	14.10	77388.6	12.45	35.1
					10	14.05	50960.1	12.68	33.1
					5	14.02	8802.5	12.55	36.8
					2	14.59	24.6	**12.42**	41.9

- When $k = 10$, Step 5 is faster for $s = 2$ or $s = 5$, which was initially unexpected. The reason is that, when k is large and s is small, the column generation optimization algorithm generates new columns heuristically, reaching the maximum allowed space of 3000 columns; thus the solution of the difficult mixed integer linear pricing subproblems is never performed. However, in those cases, poorer values of IL were obtained.
- In general, the best IL values are obtained when using Step 3, with the exception of Tarragona and $k = 10$, whose best IL was given in Table 4 without Step 3. This can be due to the randomness associated with partitioning the dataset into s subsets.
- The solution times in Step 5 are generally longer when k is large and s is small.

Finally, Table 5 summarizes the best IL results obtained with the new approach, comparing them with—to our knowledge—the best values reported in the literature by previous heuristics (citing the source), and the optimization method of [1]. The approaches implemented by those other heuristics were commented in Sect. 1 of this paper. It can be seen that the new approach provided a better solution than previous heuristics in all the cases. In addition, for $k \in \{3, 5\}$, the new approach also provided IL values close to the ones provided by the optimization method of [1], usually while requiring fewer computational resources. For instance, with our approach, the solutions for Tarragona and $k = 3$ and $k = 5$ required, respectively, 7 and 1127 s; whereas the optimization method of

Table 5. Comparison of best IL values obtained with the new approach vs. those found in the literature (citing the source).

Instance	k	New heuristic IL	Previous heuristics IL	Optimization method of [1] IL
Tarragona	3	14.50	14.77 [14]	14.46
	5	20.25	20.93 [2]	20.16
	10	**30.55**	31.95 [2]	—
Census	3	4.79	5.06 [14]	4.67
	5	7.84	8.37 [13]	7.36
	10	**12.32**	12.65 [13]	—

[1] (running on a different—likely older—computer) needed, respectively, 160 and 4779 s. For Census and $k = 3$ and $k = 5$, the solution times with our approach were, respectively, 10 and 5346 s, whereas that of [1] required, respectively, 3868 and 6788 s. The optimization method of [1] is unable to solve problems with $k = 10$, and in this case our new approach reported—as far as we know—the best IL values ever computed for these instances.

6 Conclusions

We have presented here the preliminary results from a new heuristic approach for the microaggregation problem. This method combines three ingredients: a partition of the dataset; solving each subset of the partition with an optimization-based approach; and a local search improvement heuristic. The results have shown that our new approach provides solutions that are as good as (and in some cases even better than) those reported in the literature by other heuristics for the microaggregation problem, although it may generally require longer executions times. Future research will investigate improving Step 5 of the heuristic algorithm and will further consider more sophisticated large-neighborhood search improvement heuristics.

References

1. Aloise, D., Hansen, P., Rocha, C., Santi, É.: Column generation bounds for numerical microaggregation. J. Global Optim. **60**(2), 165–182 (2014). https://doi.org/10.1007/s10898-014-0149-3
2. Aloise, D., Araújo, A.: A derivative-free algorithm for refining numerical microaggregation solutions. Int. Trans. Oper. Res. **22**, 693–712 (2015)
3. Brand, R., Domingo-Ferrer, J., Mateo-Sanz, J. M.: Reference data sets to test and compare SDC methods for protection of numerical microdata. European Project IST-2000-25069 CASC (2002). http://neon.vb.cbs.nl/casc, https://research.cbs.nl/casc/CASCtestsets.html

4. Castro, J., Gentile, C., Spagnolo-Arrizabalaga, E.: An algorithm for the microaggregation problem using column generation. Comput. Oper. Res. **144**, 105817 (2022). https://doi.org/10.1016/j.cor.2022.105817
5. Defays, D., Anwar, N.: Micro-aggregation: a generic method. In: Proceedings of Second International Symposium Statistical Confidentiality, pp. 69–78 (1995)
6. Domingo-Ferrer, J., Mateo-Sanz, J.M.: Practical data-oriented microaggregation for statistical disclosure control. IEEE Trans. Knowl. Data Eng. **14**, 189–201 (2002)
7. Domingo-Ferrer, J., Martínez-Ballesté, A., Mateo-Sanz, J.M., Sebé, F.: Efficient multivariate data-oriented microaggregation. VLDB J. **15**, 355–369 (2006)
8. Domingo-Ferrer, J., Torra, V.: Ordinal, continuous and heterogeneous k-anonymity through microaggregation. Data Mining Knowl. Disc. **11**, 195–212 (2005)
9. Ji, X., Mitchell, J.E.: Branch-and-price-and-cut on the clique partitioning problem with minimum clique size requirement. Discr. Optim. **4**, 87–102 (2007)
10. Ghosh, J., Liu, A.: K-means. In: The Top Ten Algorithms in Data Mining, pp. 21–35. Taylor & Francis, Boca Raton (2009)
11. Hansen, S., Mukherjee, S.: A polynomial algorithm for optimal univariate microaggregation. IEEE Trans. Knowl. Data Eng. **15**, 1043–1044 (2003)
12. Kaufman, L., Rousseeuw, P.J.: Partitioning around medoids (Program PAM). In: Wiley Series in Probability and Statistics, pp. 68–125. John Wiley & Sons, Hoboken (1990)
13. Khomnotai, L., Lin, J.-L., Peng, Z.-Q., Samanta, A.: Iterative group decomposition for refining microaggregation solutions. Symmetry **10**, 262 (2018). https://doi.org/10.3390/sym10070262
14. Maya-López, A., Casino, F., Solanas, A.: Improving multivariate microaggregation through Hamiltonian paths and optimal univariate microaggregation. Symmetry. **13**, 916 (2021). https://doi.org/10.3390/sym13060916
15. Oganian, A., Domingo-Ferrer, J.: On the complexity of optimal microaggregation for statistical disclosure control. Statist. J. U. N. Econ. Com. Eur. **18**, 345–354 (2001)
16. Panagiotakis, C., Tziritas, G.: Successive group selection for microaggregation. IEEE Trans. Knowl. Data Eng. **25**, 1191–1195 (2013)
17. Soria-Comas, J., Domingo-Ferrer, J., Mulero, R.: Efficient near-optimal variable-size microaggregation. In: Torra, V., Narukawa, Y., Pasi, G., Viviani, M. (eds.) MDAI 2019. LNCS (LNAI), vol. 11676, pp. 333–345. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26773-5_29
18. Spagnolo-Arrizabalaga, E.: On the use of Integer Programming to pursue Optimal Microaggregation. B.Sc. thesis, University Politècnica de Catalunya, School of Mathematics and Statistics, Barcelona (2016)
19. Solanas, A., Martínez-Ballesté, A.: V-MDAV: a multivariate microaggregation with variable group size. In: Proceedings of COMPSTAT Symposium IASC, pp. 917–925 (2006)
20. Sweeney, L.: k-anonymity: a model for protecting privacy. Int. J. Uncertain Fuzziness Knowl. Based Syst. **10**, 557–570 (2002)