SIAM J. SCI. COMPUT.
Vol. 21, No. 4, pp. 1485–1505

AN INTERIOR POINT ALGORITHM FOR MINIMUM SUM-OF-SQUARES CLUSTERING*

O. DU MERLE†, P. HANSEN‡, B. JAUMARD§, AND N. MLADENOVIĆ¶

Abstract. An exact algorithm is proposed for minimum sum-of-squares nonhierarchical clustering, i.e., for partitioning a given set of points from a Euclidean m-space into a given number of clusters in order to minimize the sum of squared distances from all points to the centroid of the cluster to which they belong. This problem is expressed as a constrained hyperbolic program in 0-1 variables. The resolution method combines an interior point algorithm, i.e., a weighted analytic center column generation method, with *branch-and-bound*. The auxiliary problem of determining the entering column (i.e., the oracle) is an unconstrained hyperbolic program in 0-1 variables with a quadratic numerator and linear denominator. It is solved through a sequence of unconstrained quadratic programs in 0-1 variables. To accelerate resolution, variable neighborhood search heuristics are used both to get a good initial solution and to solve quickly the auxiliary problem as long as global optimality is not reached. Estimated bounds for the dual variables are deduced from the heuristic solution and used in the resolution process as a trust region. Proved minimum sum-of-squares partitions are determined for the first time for several fairly large data sets from the literature, including Fisher's 150 iris.

Key words. classification and discrimination, cluster analysis, interior-point methods, combinatorial optimization

AMS subject classifications. 62H30, 90C51, 90C27

PII. S1064827597328327

1. Introduction.

Cluster analysis addresses the following general problem: Given a set of entities, find subsets, or clusters, which are homogeneous and/or well separated (Hartigan [25], Gordon [15], Kaufman and Rousseeuw [28], Mirkin [36]). This problem has many applications in engineering, medicine, and both the natural and the social sciences. The concepts of homogeneity and separation can be made precise in many ways. Moreover, a priori constraints, or in other words a structure, can be imposed on the clusters. This leads to many clustering problems and even more algorithms.

The most studied and used methods of cluster analysis belong to two categories: hierarchical clustering and partitioning. Hierarchical clustering algorithms give a hierarchy of partitions, which are jointly composed of clusters either disjoint or included one into the other. Those algorithms are agglomerative or, less often, divisive. In the first case, they proceed from an initial partition, in which each cluster contains a single entity, by successive merging of pairs of clusters until all entities are in the same one. In the second case, they proceed from an initial partition with all entities in the same cluster, by successive bipartitions of one cluster at a time until all entities are isolated, one in each cluster. The best partition is then chosen from the hierarchy of partitions obtained, usually in an informal way. A graphical representation of results,

*Received by the editors October 3, 1997; accepted for publication (in revised form) February 10, 1999; published electronically March 6, 2000. This research has been supported by the Fonds National de la Recherche Scientifique Suisse, NSERC-Canada, and FCAR-Québec.
http://www.siam.org/journals/sisc/21-4/32832.html

†GERAD, Faculty of Management, McGill University, Montréal, Canada.
‡GERAD, École des HEC, Département des Méthodes Quantitatives de Gestion, Montréal, Canada (pierreh@umontreal.ca).
§GERAD, École Polytechnique, Montréal, Canada.
¶GERAD, École des HEC, Montréal, Canada.

such as a dendrogram or an espalier (Hansen, Jaumard, and Simeone [23]), is useful for that purpose. Hierarchical clustering methods use an objective (sometimes implicit) function locally, i.e., at each iteration. With the exception of the single linkage algorithm (Johnson [27], Gower and Ross [17]) which maximizes the split of all partitions obtained (Delattre and Hansen [2]), hierarchical algorithms do not give optimal partitions for their criterion after several agglomerations or divisions. In contrast, partitioning algorithms assume given the number of clusters to be found (or use it as a parameter) and seek to optimize exactly or approximately an objective function.

Among many criteria used in cluster analysis, the minimum sum of squared distances from each entity to the centroid of the cluster to which it belongs—or minimum sum-of-squares for short—is one of the most used. It is a criterion for both homogeneity and separation as minimizing the within-clusters sum-of-squares is equivalent to maximizing the between-clusters sum-of-squares.

Both hierarchical and nonhierarchical procedures for minimum sum-of-squares clustering (MSSC) have long been used. Ward's [45] method is a hierarchical agglomerative one. It fits in Lance and Williams's [32] general scheme for agglomerative hierarchical clustering and can therefore be implemented in $O(N^2 \log N)$, where N is the number of entities considered. Moreover, using chains of near-neighbors, an $O(N^2)$ implementation can be obtained (Benzecri [1], Murtagh [38]). Divisive hierarchical clustering is more difficult. If the dimension m of the space to which the entities to be classified belong is fixed, a polynomial algorithm in $O(N^{m+1} \log N)$ can be obtained (Hansen, Jaumard, and Mladenović [22]). In practice, problems with $m = 2$, $N \leq 20000$; $m = 3$, $N \leq 1000$; $m = 4$, $N \leq 200$ can be solved in reasonable computing time. Otherwise, one must use heuristics.

Postulating a hierarchical structure for the partitions obtained for MSSC is a strong assumption. In most cases direct minimization of the sum-of-squares criterion among partitions with a given number M of clusters appears to be preferable. This has traditionally been done with heuristics, the best known of which is KMEANS [33] (see, e.g., Gordon and Henderson [16] and Gordon [15] for surveys of these heuristics). KMEANS proceeds from an initial partition to local improvements by reassignment of one entity at a time and recomputation of the two centroids of clusters to which this entity belonged and now belongs, until stability is reached. The procedure is repeated a given number of times to obtain a good local optimum.

It has long been known that entities in two clusters are separated by the hyperplane perpendicular to the line joining their centroids and intersecting it at its middle point; see, e.g., Gordon and Henderson [16]. This implies that an optimal partition corresponds to a Voronoi diagram. Such a property can be exploited in heuristics but does not lead to an efficient exact algorithm as enumeration of Voronoi diagrams is time-consuming, even in two-dimensional space (Inaba, Katoh, and Imai [26]).

Not much work appears to have been devoted, until now, to exact resolution of MSSC. The problem was formulated mathematically by Vinod [44] and Rao [39] but little was done there for its resolution. Koontz, Narendra, and Fukunaga [31] propose a branch-and-bound algorithm which was refined by Diehr [3]. Bounds are obtained in two ways: First, the sum-of-squares for entities already assigned to the same cluster during the resolution is a lower bound. Second, the set of entities to be clustered may be divided into subsets of smaller size and the sum of the sum-of-squares for each of these subsets is also a lower bound. Using these bounds for all subsets but one and assigning the entities of the last is then done. After they are assigned, the process continues with entities of the second subset and so forth. The bounds used tend not to

be very tight, and consequently, the problems solved are not very large, i.e., $N \leq 60$, with two exceptions. These are two problems with 120 entities in $\mathbb{R}^2$ belonging to two or four very well separated clusters. Such examples might not be representative of what this branch-and-bound algorithm might do on real data sets of comparable size, and it points to a difficulty in evaluating branch-and-bound algorithms by the size of the largest instance solved. Indeed, consider an example of MSSC with N entities divided into M clusters which are each within a unit ball in $\mathbb{R}^m$. Assume these balls are pairwise at least N units apart. Then any reasonable heuristic will give the optimal partition and any branch-and-bound algorithm will confirm its optimality without branching as any misclassification more than doubles the objective function value, and hence, should one be made, the bound would exceed the incumbent value. Note that N, M, and m can be arbitrarily large.

In this paper we investigate an exact algorithm for MSSC. The problem is expressed as a constrained hyperbolic program in 0-1 variables with a sum-of-ratio objective. This compact formulation is shown to be equivalent to an extended one with an exponential number of columns corresponding to all possible clusters. The resolution method combines an interior point algorithm, i.e., the weighted version of the analytic center cutting plane method (ACCPM) of Goffin, Haurie, and Vial [13] with branch-and-bound. The auxiliary problem of determining the entering column (i.e., the oracle) is an unconstrained hyperbolic program in 0-1 variables with a quadratic numerator and linear denominator. It is solved using Dinkelbach's lemma [4], by a sequence of unconstrained quadratic 0-1 programs. Moreover, to accelerate resolution, variable neighborhood search heuristics are used both to get a good initial solution and to solve quickly the auxiliary problem as long as global optimality is not reached. Estimated bounds for the dual variables are deduced from the heuristic solution and used in the resolution process as a trust region. Proved minimum sum-of-squares partitions are determined for the first time for several fairly large data sets from the literature, including Fisher's 150 iris [9].

Both the compact and the extended formulation are given in the next section, where their relationship is also studied. Basic components of our algorithm as well as several strategies are explained in section 3, while refinements, which accelerate it considerably, are discussed in section 4. Computational results and conclusions are given in section 5.

2. Model.

2.1. Compact formulation.

The MSSC problem may be formulated mathematically in several ways, which suggest different possible algorithms. We first consider a straightforward formulation.

Let $O = \{o_1, \ldots, o_N\}$ denote a set of N entities to be clustered. These entities are points in m-dimensional Euclidean space $\mathbb{R}^m$. Let $P_M = \{C_1, \ldots, C_M\}$ denote a partition of O in M classes, or clusters, i.e.,

$$C_j \neq \emptyset \quad \forall j; \quad C_i \bigcap C_j = \emptyset \quad \forall i, j \neq i; \quad \bigcup_{j=1}^{M} C_j = O.$$

Introducing binary variables x_{jk} such that

$$x_{jk} = \begin{cases} 1 & \text{if entity } o_k \text{ belongs to cluster } C_j, \\ 0 & \text{otherwise,} \end{cases}$$

the minimum sum-of-squares clustering problem may be expressed as follows:

$$
\begin{aligned}
&\text{Min} && \sum_{j=1}^{M}\sum_{k=1}^{N} x_{jk}\|o_k - z_j\|^2 && \\
&\text{subject to (s.t.)} && \sum_{j=1}^{M} x_{jk} = 1, && k = 1,\dots,N,\\
& && x_{jk} \in \{0,1\}, && j = 1,\dots,M, \quad k = 1,\dots N,
\end{aligned} \tag{1}
$$

where z_j denotes the centroid of the cluster C_j.

The algorithms of Koontz, Narendra, and Fukunaga [31] and Diehr [3] are based on such a formulation. Recall now that Huygens' theorem (e.g., Edwards and Cavalli-Sforza [8]) states that the sum of squared distances from all points of a given set to its centroid is equal to the sum of squared distances between pairs of points of this set divided by its cardinality. Hence for any cluster C_j

$$\sum_{k:o_k \in C_j} \|o_k - z_j\|^2 = \sum_{k,l:o_k,o_l \in C_j} \frac{\|o_k - o_l\|^2}{|C_j|}.$$

Therefore (1) may be written

$$
\begin{aligned}
&\text{Min} && \sum_{j=1}^{M} \frac{\displaystyle\sum_{k=1}^{N-1}\sum_{l=k+1}^{N} d_{kl}x_{jk}x_{jl}}{\displaystyle\sum_{k=1}^{N} x_{jk}} && \\
&\text{s.t.} && \sum_{j=1}^{M} x_{jk} = 1, && k = 1,\dots,N,\\
& && x_{jk} \in \{0,1\}, && j = 1,\dots,M, \quad k = 1,\dots,N,
\end{aligned} \tag{2}
$$

where $d_{kl} = \|o_k - o_l\|^2$.

This formulation was first given by Vinod [44] and Rao [39]. Moreover, one can observe that equality constraints of (2) may be replaced by

$$\sum_{j=1}^{M} x_{jk} \geq 1, \tag{3}$$

since a solution with an entity that belongs to several clusters, i.e., a covering of O which is not a partition, cannot be optimal.

The program (2) (or its variant with inequality constraint (3)) is a constrained hyperbolic program in 0-1 variables with a sum-of-ratios objective, N constraints, and $N \times M$ binary variables. It does not lead itself to an easy resolution.

2.2. Extended formulation. Partitioning problems of cluster analysis can be easily expressed by considering all possible clusters. This was already done by Rao [39]. Consider any cluster C_t and let

$$c_t = \frac{1}{|C_t|} \sum_{k,l:o_k,o_l \in C_t} d_{kl}$$

denote the sum-of-squares for C_t. Then let

$$a_{kt} = \begin{cases} 1 & \text{if entity } o_k \text{ belongs to cluster } C_t, \\ 0 & \text{otherwise.} \end{cases}$$

The extended formulation of MSSC can be written

$$\begin{array}{lll} \text{Min} & \displaystyle\sum_{t\in T} c_t y_t & \\ \text{s.t.} & \displaystyle\sum_{t\in T} a_{kt} y_t = 1, & k = 1,\dots,N, \\ & \displaystyle\sum_{t\in T} y_t = M, & \\ & y_t \in \{0,1\}, & t \in T, \end{array} \tag{4}$$

where $T = \{1,\dots,2^N - 1\}$. This is a set partitioning problem with a side constraint. Again, equality constraints in (4) can be replaced by inequalities:

$$\sum_{t\in T} a_{kt} y_t \geq 1, \qquad k = 1,\dots,N, \tag{5}$$

and

$$\sum_{t\in T} y_t \leq M. \tag{6}$$

Variables y_t are equal to 1 if cluster C_t is in the optimal partition and to 0 otherwise. The first set of constraints in (4) (resp., constraints (5)) expresses that each entity belongs to a (resp., at least one) cluster, and the next constraint in (4) (resp., constraint (6)) expresses that the optimal partition contains exactly (resp., at most) M clusters. The program (4) (resp., with inequality constraints (5) and (6)) is a large linear partitioning (resp., covering) problem with one additional constraint on the number of variables at 1. The number of variables is exponential in the number N of entities. Therefore this problem cannot be written explicitly and solved in a straightforward way unless N is small. Fortunately, as shown below, the column-generation technique of linear programming (Gilmore and Gomory [10]) can be used together with branch-and-bound to solve exactly large instances.

2.3. Relationship between formulations. There is a close relationship between the two formulations of MSSC. To show this, consider the Lagrangian relaxation of the compact formulation:

$$\max_{\lambda_k \geq 0} \quad \min_{x_{jk}\in\{0,1\}} \left(\sum_{j=1}^{M} \frac{\displaystyle\sum_{k=1}^{N-1} \sum_{l=k+1}^{N} d_{kl} x_{jk} x_{jl}}{\displaystyle\sum_{k=1}^{N} x_{jk}} - \sum_{k=1}^{N} \sum_{j=1}^{M} \lambda_k (x_{jk} - 1) \right), \tag{7}$$

where the λ_k are the Lagrange multipliers. As, in absence of constraints, the minimum of a sum is equal to the sum of minima of its terms, problem (7) is equivalent to

$$\max_{\lambda_k \geq 0} \left(\sum_{k=1}^{N} \lambda_k + \sum_{j=1}^{M} \min_{x_{jk}\in\{0,1\}} \left(\frac{\displaystyle\sum_{k=1}^{N-1} \sum_{l=k+1}^{N} d_{kl} x_{jk} x_{jl}}{\displaystyle\sum_{k=1}^{N} x_{jk}} - \sum_{k=1}^{N} \lambda_k x_{jk} \right) \right). \tag{8}$$

Observe that index j does not appear in the data of the minimization. Hence (8) can be written after removing j:

$$\max_{\lambda_k \geq 0} \left(\sum_{k=1}^{N} \lambda_k + M f(\lambda) \right), \tag{9}$$

where

$$f(\lambda) = \min_{x_k \in \{0,1\}} \left(\frac{\sum_{k=1}^{N-1} \sum_{l=k+1}^{N} d_{kl} x_k x_l}{\sum_{k=1}^{N} x_k} - \sum_{k=1}^{N} \lambda_k x_k \right). \tag{10}$$

Several remarks are in order. First $f(\lambda)$ is the lower envelope of a set of linear functions corresponding to all possible values of x. Hence it is piecewise linear and concave. Second, $f(\lambda)$ is nonpositive, as the expression to be minimized in (10) is equal to $-\lambda_1$ for $x_1 = 1$, $x_k = 0$, $k \neq 1$, and $\lambda_1 \geq 0$. Third, this expression can be written as the ratio of a quadratic function to a linear function in 0-1 variables (using the fact that $x_k^2 = x_k$ for binary variables):

$$f(\lambda) = \min_{x_k \in \{0,1\}} \frac{\sum_{k=1}^{N-1} \sum_{l=k+1}^{N} (d_{kl} - \lambda_k - \lambda_l) x_k x_l - \sum_{k=1}^{N} \lambda_k x_k}{\sum_{k=1}^{N} x_k}. \tag{11}$$

Fourth, the optimal value of (9) is not larger than that of the compact formulation (2) of which it is a relaxation (see, e.g., Minoux [35]).

From concavity of $f(\lambda)$ one has

$$f(\lambda) \leq f(\lambda^t) + \sum_{k=1}^{N} g_k^t (\lambda_k - \lambda_k^t) \tag{12}$$

for any vector $\lambda^t \geq 0$, where $g^t \in \partial f(\lambda^t)$, i.e., g^t is a subgradient of f at λ^t. In our case one such subgradient is $-x^t$, where x^t is an (not necessarily unique) optimal solution of (10) for λ^t. Moreover,

$$f(\lambda^t) - \sum_{k=1}^{N} g_k^t \lambda_k^t = \frac{\sum_{k=1}^{N-1} \sum_{l=k+1}^{N} d_{kl} x_k^t x_l^t}{\sum_{k=1}^{N} x_k^t}$$

which is equal to c_t, the cost of the cluster C_t defined by x^t. Hence (12) is one of the hyperplanes defining $f(\lambda)$. Let T' denote the index set of vectors x^t corresponding to

all such hyperplanes. Then (9) may be written

$$\begin{array}{lll} \text{Max} & Mz + \sum_{k=1}^{N} \lambda_k, & \\ \text{s.t.} & z + \sum_{k=1}^{N} x_k^t \lambda_k \le c_t & \forall t \in T', \\ & \lambda_k \ge 0, & k = 1, \ldots, N, \\ & z \le 0. & \end{array} \tag{13}$$

The dual of this linear program, with dual variables denoted by y_t, is

$$\begin{array}{lll} \text{Min} & \sum_{t \in T'} c_t y_t & \\ \text{s.t.} & \sum_{t \in T'} x_k^t y_t \ge 1, & k = 1, \ldots, N, \\ & \sum_{t \in T'} y_t \le M, & \\ & y_t \ge 0, & t \in T', \end{array} \tag{14}$$

and setting

$$a_{kt} = \begin{cases} 1 & \text{if } x_k^t = 1, \\ 0 & \text{otherwise,} \end{cases}$$

one sees that (14) is equal to the linear relaxation of the extended formulation (4) except for the fact that variables with index $t \in T \backslash T'$ have been deleted. Such variables correspond to redundant constraints in (13). Indeed, assume this is not the case. Then there exists $\bar{\lambda} \in \mathbb{R}_+^n$, $\bar{z} \in \mathbb{R}_-$, x^r, and c_r with $r \in T \backslash T'$ such that

$$\bar{z} + \sum_{k=1}^{N} x_k^t \bar{\lambda}_k \le c_t \qquad \forall t \in T'$$

and

$$\bar{z} + \sum_{k=1}^{N} x_k^r \bar{\lambda}_k > c_r.$$

This implies, replacing c_r by its value, that

$$\bar{z} > \frac{\sum_{k=1}^{N-1} \sum_{l=k+1}^{N} d_{kl} x_k^r x_l^r}{\sum_{k=1}^{N} x_k^r} - \sum_{k=1}^{N} x_k^r \bar{\lambda}_k. \tag{15}$$

As the right-hand side of (15) is the objective of the minimization problem (10) in which variables x_k are fixed at x_k^r, it is larger than or equal to this minimum $f(\bar{\lambda})$, contradicting $z \le f(\lambda) \forall \ge 0$.

We have thus shown that the Lagrangian relaxation of the compact formulation is equivalent to the linear relaxation of the extended formulation. This equivalence will be exploited in the algorithm of the next section.

From now on program (14) will be referred to as the primal, while program (13) will be called the dual, in order to conform with usual convention in the description of column generation algorithms.

3. Algorithm. The extended formulation of MSSC, i.e., problem (4), will be solved by combining a column generation procedure with branch-and-bound. In the dual, resolution of the continuous relaxation of problem (4) can be viewed as a cutting-plane, or outer-approximation, method applied to solve problem (9). Cuts are iteratively added to the restricted dual, i.e., one considers an increasing subset of rows of (13) or equivalently, an increasing subset of columns of (14). Strategies for solving this problem are discussed in the following subsection. The auxiliary problem of finding the cuts (or columns in the primal) is examined in subsection 3.2. Branching is discussed in subsection 3.3. Several refinements, i.e., finding an initial solution, using it to bound the dual variables, and weighting appropriately constraints in the cutting-plane method to ease resolution of the auxiliary problem, will be described in the next section.

3.1. Strategies for solving the linear relaxation. We consider three strategies for solving the Lagrangian relaxation of MSSC. They correspond to different ways of choosing the Lagrange multipliers λ. For all three strategies the stopping criterion is defined by an ε relative duality gap where epsilon is a small positive number (e.g., 10^{-6}). Lower bounds θ_l are given by the value of the Lagrangian problem, i.e., for a given λ

$$\theta_l = \sum_{k=1}^{N} \lambda_k + Mf(\lambda).$$

Upper bounds θ_u vary with the strategy and will be given below. The relative duality gap is defined as $\frac{\theta_u - \theta_l}{\max(\theta_l, 1)}$.

A first strategy is that of Kelley [29] in which at each iteration λ is chosen as the optimal solution of the current restricted dual. This corresponds, viewing the method as a column generation one, to the minimum reduced cost criterion of the simplex algorithm in the primal. It is the most frequently used strategy in column-generation algorithms. However, it suffers from very slow convergence for the MSSC problem. Indeed, as solutions to this problem are massively primal degenerate, after an optimal solution has been found many further iterations may be needed to prove its optimality.

In this case, the upper bound θ_u on the optimal value of the Lagrangian relaxation (9) is given by the optimal value of the restricted dual (13) (or of the restricted primal (14)).

A second strategy aims at stabilizing the column generation procedure, using a bundle method in l_1 norm (Hansen et al. [21], du Merle et al. [7]). A linear penalization term is then added to the current restricted dual:

$$\begin{aligned} \lambda^{s+1} = \text{Arg Max} \quad & Mz + \sum_{k=1}^{N} \lambda_k - \mu \sum_{k=1}^{N} |\lambda_k - \tilde{\lambda}_k|, & \\ \text{s.t.} \quad & z + \sum_{k=1}^{N} x_k^t \lambda_k \le c_t, & t = 1, \ldots, s, \\ & \lambda_k \ge 0, & k = 1, \ldots, N, \\ & z \le 0, & \end{aligned} \tag{16}$$

where $\tilde{\lambda}$ is the best known value for λ and μ is a parameter. Problem (16) can be

reformulated as a linear program by addition of $2N$ new variables and N constraints:

$$
\begin{aligned}
\lambda^{s+1} = \text{Arg Max} \quad & Mz + \sum_{k=1}^{N} \lambda_k - \mu \sum_{k=1}^{N} (u_k + v_k), & \\
\text{s.t.} \quad & z + \sum_{k=1}^{N} x_k^t \lambda_k \le c_t, & t = 1, \ldots, s, \\
& \lambda_k + u_k - v_k = \tilde{\lambda}_k, & k = 1, \ldots, N, \\
& \lambda_k, u_k, v_k \ge 0, & k = 1, \ldots, N, \\
& z \le 0. &
\end{aligned} \tag{17}
$$

Under this form the penalized restricted dual is a parametric linear program. In order to ensure convergence of this penalized program to the optimal solution of the original problem, μ is progressively decreased to 0. Changes in μ are made each time the duality gap of the penalized problem is less than ε. Similar to the first strategy, the upper bound on the optimal value of the penalized problem is given by the optimal value of (17). Again, as in the first strategy, computations take place in the primal, which has less rows than columns.

In the third strategy, computations take place in the dual and an interior point method, i.e., the ACCPM of Goffin, Haurie, and Vial [13] is used. In this strategy, Lagrangian multipliers are solutions of the following problem:

$$
\begin{aligned}
\lambda^{s+1} = \text{Arg Max} \quad & \sum_{t=1}^{N+s+1} \mu_t \log s_t, & \\
\text{s.t.} \quad & z + \sum_{k=1}^{N} x_k^t \lambda_k + s_t = c_t, & t = 1, \ldots, s, \\
& \lambda_k - s_{k+s} = 0, & k = 1, \ldots, N, \\
& z + s_{N+s+1} = 0, & \\
& Mz + \sum_{k=1}^{N} \lambda_k + s_0 = \theta_l, &
\end{aligned} \tag{18}
$$

where the parameters μ_t are weights given to the cuts and the s_t are slack variables for these cuts (including nonnegativity and nonpositivity constraints). The solution of this problem is a weighted analytic center for the current set of constraints. Several ways of choosing the weights μ_t are considered. If they are all equal to 1, the standard analytic center is obtained. In practice, this choice is known to be less efficient than one in which $\mu_0 = N + s + 1$, the other μ_t remaining at 1. The effect is to push the analytic center away from the lower bound constraint [6]. A third choice will be discussed in the next section.

Here the upper bound θ_u on the optimal solution of the Lagrangian relaxation (9) is a by-product of the resolution of problem (18), which ends up with a primal and dual feasible point. Details on efficient computations of this bound are given in [5].

3.2. Auxiliary problem. The auxiliary problem is to find the value of $f(\lambda^s)$ and a subgradient of f at λ^s. As shown in (11) above, finding $f(\lambda^s)$ can be expressed as a hyperbolic (or fractional) program in 0-1 variables with quadratic numerator and linear denominator. This is done both in a heuristic and in an exact way. The heuristic will be discussed in the next section.

The standard way to solve such problems (see, e.g., Schaible [42]) is to use Dinkelbach's lemma [4]. It works as follows:

Initialization. Let f_0 be the value of an initial solution x^0 (finding a good initial solution will be discussed in the next section). One can always set $f_0 = 0$. Set $i \leftarrow 1$;

Repeat the following until the stopping condition is met:

(1) Solve

$$x^i = \underset{x_k \in \{0,1\}}{\text{Arg Min}} \sum_{k=1}^{N-1} \sum_{l=k+1}^{N} (d_{kl} - \lambda_k^s - \lambda_l^s) x_k x_l - \sum_{k=1}^{N} (\lambda_k^s + f_{i-1}) x_k; \tag{19}$$

(2) If the optimal value of (19) is smaller than $-\varepsilon$, where ε is a small positive constant (i.e., 10^{-6}),

$$f_i = \frac{\displaystyle\sum_{k=1}^{N-1} \sum_{l=k+1}^{N} (d_{kl} - \lambda_k^s - \lambda_l^s) x_k^i x_l^i - \sum_{k=1}^{N} \lambda_k^s x_k^i}{\displaystyle\sum_{k=1}^{N} x_k^i},$$

set $i \leftarrow i + 1$ and return to the previous step. Otherwise, stop.

Convergence of this algorithm follows from a general result of Megiddo [34]. Within this algorithm, however, it is necessary to solve at each iteration the unconstrained quadratic 0-1 program (19). The exact algorithm to solve this problem is a branch-and-bound method which exploits the following:

- Representation of the function as a posiform, i.e., as a polynomial in variables x_k and $\bar{x}_k = 1 - x_k$ with positive coefficients only (except possibly for a constant). Moreover, the posiform for which the constant term is largest is obtained by removing variables from the linear terms in x_k and $\bar{x}_k$ (as $x_k + \bar{x}_k = 1$) and by considering chains of complementation increasing this constant (e.g., $x_k + \bar{x}_k x_l + \bar{x}_l = 1 + x_k \bar{x}_l$) (Hammer, Hansen, and Simeone [19]).
- First-order Boolean derivatives (e.g., Hammer and Hansen [18]):

$$\delta_{x_k} = \sum_{l=1}^{k-1} (d_{lk} - \lambda_l^s - \lambda_k^s) x_l + \sum_{l=k+1}^{N} (d_{kl} - \lambda_k^s - \lambda_l^s) x_l - \lambda_k^s - f_{i-1} \tag{20}$$

and

$$\delta_{x_k} \geq 0 \Rightarrow x_k = 0,$$
$$\delta_{x_k} < 0 \Rightarrow x_k = 1.$$

- Branching on the derivatives (i.e., $\delta_{x_k} \geq 0$ and $\delta_{x_k} < 0$) and fixing x_k at 0 or 1 as a consequence (instead of branching on the variables x_k as is usually done). This gives linear relations in variables x_l which can be used to fathom the current subproblem by showing it contains no local optimum.

Remark. Within Dinkelbach's algorithm one could solve the unconstrained quadratic 0-1 programming problem by some heuristic as long as the stopping condition is not verified. Then, however, one must use an exact algorithm to guarantee the optimality of x_i and one must iterate if the stopping condition no longer holds.

3.3. Branching rule. As the solution of the continuous relaxation of the extended formulation (4) is not necessarily integer, a branching rule must be specified. We use the (now) standard one in column generation, due to Ryan and Foster [41]. It consists of finding two rows i_1 and i_2 such that there are two columns t_1 and t_2 with fractional variables at the optimum and such that

$$a_{i_1 t_1} = a_{i_2 t_1} = 1$$

(21) and

$$a_{i_1 t_2} = 1, \quad a_{i_2 t_2} = 0.$$

Such rows and columns necessarily exist in any noninteger optimal solution of the continuous relaxation of a partitioning problem [41]. While we solve a covering problem (with one additional constraint) such an optimal solution satisfies the constraints as equalities and hence this result still holds.

This branching is done by imposing on the one hand the constraint

$$x_{i_1} = x_{i_2}, \tag{22}$$

i.e., both entities will be in the same cluster, and on the other hand

$$x_{i_1} + x_{i_2} \leq 1, \tag{23}$$

i.e., the two entities cannot be in the same cluster. (Of course, in both subproblems there will be clusters containing neither of the entities, i.e., one may have $x_{i_1} = x_{i_2} = 0$.) These constraints could be added to the master problem, but it is more efficient to remove the columns in the current reduced master problem which do not satisfy them and then introduce them in the auxiliary problem. The effect in the auxiliary problem of constraint (22) is to reduce by one the number of variables and update coefficients, while constraint (23) can be imposed by setting $d_{i_1 i_2}$ to an arbitrary large value. Hence the form of the auxiliary problem is unchanged.

The branching rule so defined can be made more precise by choosing from the pairs of rows and columns satisfying condition (21) that one which is best according to another criterion, e.g., having fractional variables the closest possible to 1/2. Such refinements have not been investigated in detail as for MSSC branching appears to be rare in practice and the resulting trees are very small.

Note that the branching rule described above can also be viewed as one for the compact formulation (2), to which one adds the constraint (22) or (23). Then the first possibility mentioned above corresponds to dualizing these constraints in addition to the inequality constraints (3) while the second amounts to not doing so, hence having them as constraints in the auxiliary problem (10) obtained when relaxing problem (2).

Note also that instead of removing the columns which violate the branching constraint or giving them a large value in the objective function, one may modify them by removal or addition of an entity to satisfy his constraint and update their value in the objective function. This is due to the fact that all clusters are a priori feasible.

4. Refinements.

4.1. Heuristics. As already mentioned, heuristics can be used in several places in order to accelerate the algorithm whose principles have been described in the previous section. In some cases, such as when using Kelley's strategy, the effect of such

heuristics is quite drastic. They can be used to (i) find an initial solution of problem (2), (ii) find a solution of the auxiliary problem (11), and, possibly, (iii) find a solution of the unconstrained quadratic 0-1 program (19) which arises when solving (11).

For these problems we use a recent, simple and effective metaheuristic (or framework for heuristics) called variable neighborhood search (VNS)(Mladenović and Hansen [37], Hansen and Mladenović [24]).

The principle of VNS is to change and randomly explore neighborhoods with an appropriate local search routine. Contrary to other metaheuristics, e.g., simulated annealing (Kirkpatrick, Gelatt, and Vecchi [30]) or Tabu search (Glover [11, 12]), VNS does not follow a trajectory but explores increasingly distant neighborhoods of the current incumbent solution, and jumps from there to a new one if and only if an improvement has been made, through the local search. In this way, often favorable characteristics of the incumbent solution, e.g., that many variables are already at their optimal value, will be kept and used to obtain promising neighboring solution.

Let us denote a finite set of preselected neighborhood structures with $\mathcal{N}_k$ ($k = 1, \ldots, k_{max}$) and with $\mathcal{N}_k(x)$ the set of solutions in the k^{th} neighborhood of x. Steps of a basic VNS heuristic, applied to the problem $\min\{f(x) : x \in S\}$, are the following:

Initialization. Select the set of neighborhood structures $\mathcal{N}_k$, $k = 1, \ldots, k_{max}$, that will be used in the search; find an initial solution x; choose a stopping condition;

Repeat the following until the stopping condition is met:

(1) Set $k \leftarrow 1$;

(2) Until $k = k_{max}$, repeat the following steps:

(a) *Shaking.* Generate a point x' at random from the kth neighborhood of x ($x' \in \mathcal{N}_k(x)$);

(b) *Local search.* Apply some local search method with x' as initial solution; denote with x'' the so-obtained local optimum;

(c) *Move or not.* If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with $\mathcal{N}_1$ ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$;

Note that VNS uses only one parameter k_{max} (except for computer time allocated, e.g., the stopping condition), which can often be disposed of, e.g., by setting it equal to the size of the vector x considered.

For all the problems considered the neighborhood structure $\mathcal{N}_k(x)$ is defined by the Hamming distance ρ between solutions x and x' (i.e., the number of components in which these vectors differ):

$$\rho(x, x') = k \Leftrightarrow x' \in \mathcal{N}_k(x).$$

The local search routine is a greedy algorithm on the neighborhood $\mathcal{N}_1(x)$. Updating of first-order Boolean derivatives is used to accelerate computations (Hansen, Jaumard, and da Silva [20]).

It appears that, while this is neither required nor proved in those steps, VNS very often gives the optimal solution for MSSC as well as for the auxiliary problem in moderate computing time.

4.2. Bounds on dual variables. Let $l_k \geq 0$ (resp., h_k) be lower bounds (resp., upper bounds) on the dual variables λ_k, $k = 1, \ldots, N$. The dual problem (13) is then

$$\begin{array}{lll} \text{Max} & Mz + \sum_{k=1}^{N} \lambda_k, & \\ \text{s.t.} & z + \sum_{k=1}^{N} x_k^t \lambda_k \leq c_t & \forall t \in T', \\ & l_k \leq \lambda_k \leq h_k, & k = 1, \ldots, N, \\ & z \leq 0 & \end{array} \tag{24}$$

and its primal

$$\begin{array}{lll} \text{Min} & -\sum_{k=1}^{N} l_k \xi_k + \sum_{k=1}^{N} h_k \eta_k + \sum_{t \in T'} c_t y_t, & \\ \text{s.t.} & -\xi_k + \eta_k + \sum_{t \in T'} x_k^t y_t \geq 1, & k = 1, \ldots, N, \\ & \sum_{t \in T'} y_t \leq M, & \\ & y_t, \xi_k, \eta_k \geq 0, & t \in T', k = 1, \ldots, N, \end{array} \tag{25}$$

where ξ_k (resp., η_k) are dual variables associated to the constraints $l_k \leq \lambda_k$ (resp., $\lambda_k \leq h_k$), $k = 1, \ldots, N$.

In order to estimate the lower bound $l_{k'}$ let us first assume that only $\lambda_{k'}$ is upper bounded (e.g., $0 \leq \lambda_{k'} \leq h_{k'}$ and $0 \leq \lambda_k$ for $k = 1, \ldots, N$ and $k \neq k'$). Then observe that if $\eta_{k'} > 0$, this implies that $\lambda_{k'} \leq h_{k'}$ is active and thus if $l_{k'} \leq h_{k'}$, $l_{k'}$ is a lower bound of $\lambda_{k'}$. Therefore, to find a lower bound of $\lambda_{k'}$ we look for values of $h_{k'}$ that imply that $\eta_{k'} > 0$.

Let $v(O)$ be the value of the Lagrangian relaxation (9) of problem (2) when we are looking for a partition of all entities of O in M clusters. If

$$h_{k'} + v(O \backslash \{o_{k'}\}) < v(O),$$

then $\eta_{k'}$ in (25) will be strictly positive at the optimum. This implies that

$$l_{k'} < v(O) - v(O \backslash \{o_{k'}\}).$$

To estimate $v(O) - v(O \backslash \{o_{k'}\})$ we bound $v(O \backslash \{o_{k'}\})$ from above. Let $v_{int}(O)$ be the value of the initial solution found by VNS and let $\bar{c}_1, \ldots, \bar{c}_M$ denote the cost of the different clusters ($v_{int}(O) = \sum_{j=1}^{M} \bar{c}_j$). Let $\tilde{k}$ be the index of the cluster in the initial solution that contains $o_{k'}$ and $\tilde{c}_{\tilde{k}}$ the cost of this cluster when we omit $o_{k'}$. We thus have an initial solution and its value for problem (2) when we omit entity $o_{k'}$ in O.

$$v_{int}(O \backslash \{o_{k'}\}) = \sum_{j=1,\ j \neq \tilde{k}}^{M} \bar{c}_j + \tilde{c}_{\tilde{k}} = v_{int}(O) - \bar{c}_{\tilde{k}} + \tilde{c}_{\tilde{k}}.$$

Noting that $v(O \backslash \{o_{k'}\})$ is the value of the Lagrangian relaxation of problem (2) when we are looking for a partition of $O \backslash \{o_{k'}\}$ in M clusters, it follows that

$$l_k < v(O) - v_{int}(O) + \bar{c}_{\tilde{k}} - \tilde{c}_{\tilde{k}} \leq v(O) - v(O \backslash \{o_{k'}\}).$$

The value of $v(O)$ being unknown, it has to be estimated. As VNS often finds the optimal solution of (2) and again often there is no integrality gap, the difference $v(O) - v_{int}(O)$ was considered as 0 and $l_{k'}$ set to $\bar{c}_{\tilde{k}} - \tilde{c}_{\tilde{k}}$. This implies that one must check that the constraint $l_{k'} \leq \lambda_{k'}$ is not active at the optimum of the Lagrangian relaxation (9). Should it be active, the bound $l_{k'}$ must be reduced and the resolution of (9) resumed.

The procedure just described shows how to evaluate the lower bounds on λ_k one at a time. The values so obtained are still valid when all l_k are put together. Again, because $v(O) - v_{int}(O)$ is assumed to be null, we must check for bounds which are tight at the optimum, modify them, and resume the resolution if so.

Similarly, one can show that

$$h_{k'} > v(O \cup \{o_{k'}\}) - v(O),$$

where $v(O \cup \{o_{k'}\})$ denotes the value of the Lagrangian relaxation of a partition of O with two copies of $o_{k'}$ into M clusters. Then noting that

$$v(O \cup \{o_{k'}\}) < v_{int}(O) + \hat{c}_j - \bar{c}_j,$$

where $\hat{c}_j$ $j \neq \tilde{k}$ is the cost of the cluster C_j to which $o_{k'}$ is added, to minimize the right-hand side one chooses C_j such that the resulting $\hat{c}_j - \bar{c}_j$ is minimum. Hence

$$h_{k'} > v_{int}(O) - v(O) + \hat{c}_j - \bar{c}_j \geq v(O \cup \{o_{k'}\}) - v(O).$$

As before, we assume that $v_{int}(O) - v(O)$ is equal to 0 and add a procedure to check if the bound $h_{k'} = \hat{c}_j - \bar{c}_j$ is active at the optimum.

4.3. Weights on cuts. The most time-consuming step of the algorithm is the resolution of the auxiliary problem through a sequence of unconstrained quadratic 0-1 programming problems. As long as the relative duality gap is greater than ε the auxiliary problem (11) is solved by the VNS heuristic, which is fairly fast. However, when no relative duality gap is observed, one must check if the dual variables are optimal and therefore solve the auxiliary problem with the exact branch-and-bound algorithm. For large instances, this algorithm is time-consuming and its use, in fact, limits the size of problems which can be solved exactly.

One may note that there is some flexibility in the way the coefficients in the subproblem, i.e., the dual variables at the current iteration, are selected. Indeed, as the optimal solution, in which the number M of clusters is usually much smaller than the number N of entities, is massively primal degenerate ($N + 1 - M$ basic variables being equal to 0 when this optimal solution is integer) there is a large polytope of optimal solutions in the dual. To prove optimality of the linear relaxation of (4) (or equivalently of problem (9)) has been attained one must check the value of the lower bound θ_l. The question is then, "Which values should be selected in this polytope to make the resolution of (19) the easiest possible?"

As the branch-and-bound algorithm relies on the first-order Boolean derivatives (20), one may look for values which will make them fixed variables as soon as possible. Observe then that reducing λ_k makes condition $\delta_{x_k} \geq 0 \Rightarrow x_k = 0$ more likely to hold. (This augments the constant term as well as all coefficients of the linear terms in (20).) To achieve this goal with ACCPM we weight the upper bound constraints $\lambda_k \leq h_k$ by a constant much larger than for the other constraints. (In practice a value of 100 for the upper bound constraints and 1 for the other constraints, except for the lower bound on the problem value for which we keep the value $\mu_0 = N + s + 1$, gives good results.)

Rules are next given

Initialization:
  Set the restricted dual to an empty problem;
  Compute an initial partition using VNS;
    Add the corresponding column to the restricted dual;
    Compute bounds on dual variables using this solution;
  $t \leftarrow 0$, stop $\leftarrow$ false
REPEAT
  $\theta_u = +\infty$; $\tilde{\theta}_l = -\infty$
  REPEAT
    $(\lambda_t, \tilde{\theta}_u) \leftarrow$ `SolveTheRestrictedDual`(x^t, c^t);
    $(x^t, c^t, \theta_l) \leftarrow$`SolveTheAuxiliaryProblemUsingVNS`$(\lambda_t)$;
    $t \leftarrow t + 1$
  WHILE $\frac{\theta_u - \tilde{\theta}_l}{\max(\tilde{\theta}_l, 1)} > \varepsilon$
  $(x^t, c^t, \theta_l) \leftarrow$ `SolveTheAuxiliaryProblemUsingDinkelbach`(λ_t);
  `TestBoundOnDualVariables`();
  IF $(\theta_l = \tilde{\theta}_l$ AND `NoBoundPushed`$())$ THEN stop $\leftarrow$ true
WHILE $\neg$ stop

FIG. 1. *Algorithm to compute the lower bounds.*

4.4. Algorithm to compute the lower bounds. We state here the complete algorithm to compute the lower bounds in the branch-and-bound algorithm (see Figure 1). The following procedures (in typewriter style in the algorithm) need to be defined:

- $(\lambda_t, \tilde{\theta}_u) \leftarrow$`SolveTheRestrictedDual`$(x^t, c^t)$ adds the new cluster (x^t, c^t) to the restricted dual and solves it using the strategy chosen. It returns the upper bound θ_u and the Lagrangian multipliers λ_t.
- $(x^t, c^t, \theta_l) \leftarrow$`SolveTheAuxiliaryProblemUsingVNS`$(\lambda_t)$ solves the auxiliary problem for the Lagrangian multipliers λ_t using the heuristic VNS. It returns the cluster (x^t, c^t) and the lower bound θ_l.
- $(x^t, c^t, \theta_l) \leftarrow$`SolveTheAuxiliaryProblemUsingDinkelbach`$(\lambda_t)$ is the same as the procedure `SolveTheAuxiliaryProblemUsingVNS`(), but using Dinkelbach's algorithm and quadratic 0-1 programming.
- `TestBoundOnDualVariables`() tests if the bounds on the dual variables are active and pushes them if so.
- `NoBoundPushed`() answers true if no bounds have been pushed during the last call of `TestBoundOnDualVariables`(), no otherwise.

These rules are embedded in a standard branch-and-bound procedure, with the branching rule described above.

5. Computational results. The algorithm has been coded in C except for VNS heuristics, coded in FORTRAN. Resolution of linear programs was done with

CPLEX 3.0, and for solving the convex problem (18) we used ACCPM's library [14], slightly modified to incorporate a branch-and-bound scheme and to permit setting of appropriate weights. Results were obtained using a SUN ULTRA 200 MHz station with g++ (Option -O4) C compiler and f77 (Option -O4 -xcg92) FORTRAN compiler.

Extensive comparisons were made using some standard problems from the cluster analysis literature (Ruspini's 75 points in the Euclidean plane [40], Späth's 89 postal zones problem in three dimensions [43], and Fisher's celebrated 150 iris problem in four dimensions [9]).

We next comment in detail on results for Ruspini's data. As most of the time is spent in the resolution of the auxiliary problem, it is fair to compare resolution strategies by their number of iterations for solving the linear relaxation. These numbers for $M = 2$ to 10 and 15, 20, 25, 30, 35, and 40 are represented in Figures 2 and 3. For the results of Figure 2 no information was used, i.e., no initial solution nor bounds on the dual variables. As often observed when using column generation, Kelley's strategy is not efficient and the cases $M = 2$ and 3 could not be solved. The bundle method in l_1-norm allows solution of these two cases but is not much quicker than Kelley's strategy for M large. ACCPM is the most efficient strategy for all M and particularly for the more realistic case of M small. The number of iterations is always small, e.g., about 100 for $M = 40$ a case where at least 40 columns must be generated.

Information was exploited for the results of Figure 3, i.e., an initial heuristic solution was found by VNS, bounds on the dual variables were computed, and the columns corresponding to the clusters of the heuristic solution as well as the modified clusters defined when computing the dual bounds were added. The reduction in number of iterations due to this information is drastic for all strategies and all values of M. With Kelley's strategy all instances can then be solved. Ranking of strategies remains the same as in the former case. The number of iterations of ACCPM for M small is few and never exceeds 25. It is often smaller than M.

Figure 4 shows the effect of information on Ruspini's problem with the best strategy (i.e., ACCPM). The first plot, "version 1," gives the number of iterations when no information was used. The second plot, "version 2," gives the number of iterations when the clusters found with the initial heuristic are added at the outset. In the third plot, "version 3," we initially add also the clusters obtained during the computation of the dual bounds, and in the last one, "version 4," we also add those bounds.

The effect of the weights on the upper bounds are displayed in Table 1 using Fisher's iris data, ACCPM, and initial information. The first column gives values of M; the second, the number of iterations for solving the Lagrangian relaxation; the third, the time spent in the auxiliary problem when solved exactly with Dinkelbach's algorithm; and the fourth, the total CPU time. The next three columns give the same information as columns 2 to 4 but using a weighted version of ACCPM instead of the standard one. Times are given in seconds. Observe that the number of iterations to compute the Lagrangian relaxation with the weighted version of ACCPM is larger than in the standard version, a consequence of looking for small values for λ_k. However, the reduction of the time for solving the auxiliary problem more than compensates for this increase for the more realistic case of M small.

Computational results using a weighted version of ACCPM and initial information for the problems of Ruspini [40], Späth [43], and Fisher [9] are given in Tables 2–4. The first column gives values of M; the second, optimal solutions; the third, number of iterations for solving the Lagrangian relaxation and when needed, in parentheses,

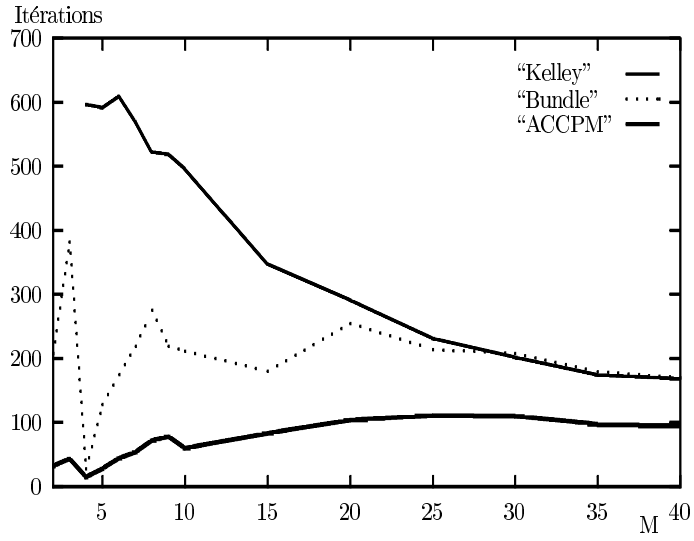

FIG. 2. *Strategies without information.*

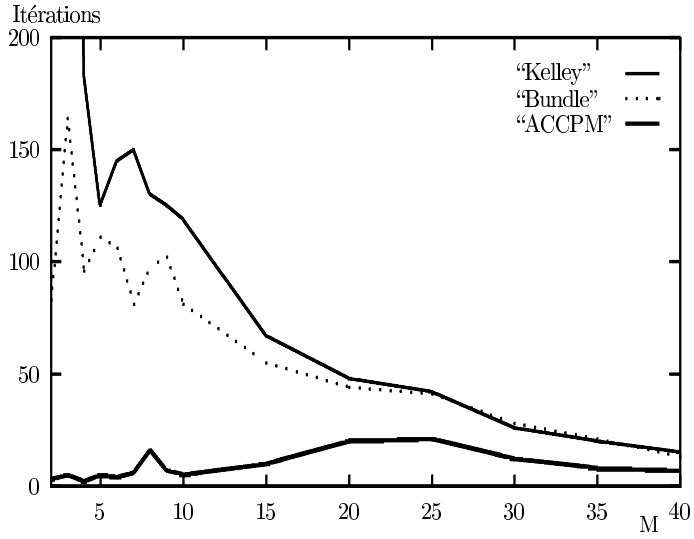

FIG. 3. *Strategies with information.*

the number of nodes in the branch-and-bound tree; the fourth, the CPU time to compute the analytic center; the fifth, the time spent in the auxiliary problem; and the last one, the total CPU time. Again, times are given in seconds. Observe again that the number of iterations to compute the Lagrangian relaxation with the weighted version of ACCPM is slightly larger than in Figures 3 and 4 obtained with a standard ACCPM.

It appears that the proposed interior point algorithm, together with exploitation of information provided by VNS, allows exact resolution of a substantially larger realistic MSSC problem than done before. The number of iterations remains small

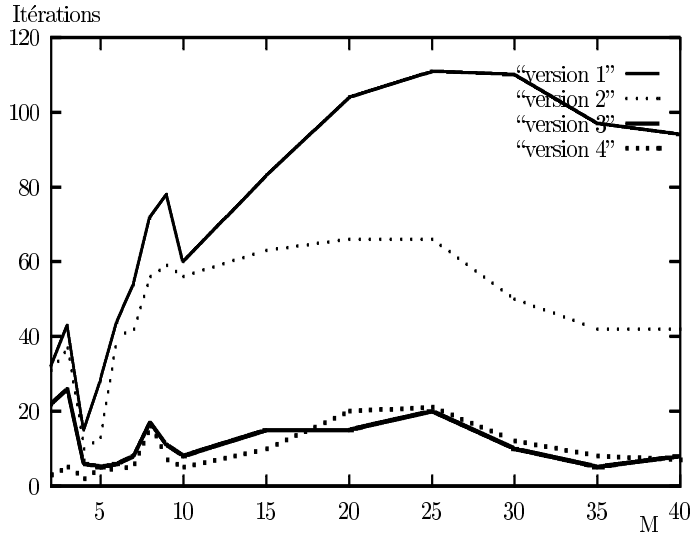

FIG. 4. *Effect of information on ACCPM.*

TABLE 1
Effect of weights on upper bounds.

Fisher	Standard ACCPM			Weighted ACCPM		
M	# of iter.	Aux. time	Tot. time	# of iter.	Aux. time	Tot. time
2	4	128313.93	128348.50	19	18190.98	18348.09
3	5	29266.86	29308.10	39	191.14	498.58
4	9	2067.87	2139.32	51	108.59	506.77
5	8	218.76	281.95	42	20.27	350.81

TABLE 2
Ruspini using weighted ACCPM.

M	Opt. sol.	# of iter.	AC time	Aux. time	Tot. time
2	89337.8	11	1.42	10.84	12.33
3	51063.4	16	.98	15.46	16.50
4	12881.0	7	.41	6.82	7.30
5	10126.7	14	.46	13.89	14.43
6	8575.40	25	.76	24.59	25.45
7	7126.19	31	.60	29.42	30.19
8	6149.63	(3) 44	1.03	41.88	43.11
9	5181.65	31	.63	30.47	31.26
10	4446.28	28	.51	26.99	27.62

TABLE 3
Späth using weighted ACCPM.

M	Opt. sol.	# of iter.	AC time	Aux. time	Tot. time
2	$6.02546\ 10^{11}$	4	2.31	17.50	19.92
3	$2.94506\ 10^{11}$	10	4.41	1475.18	1479.75
4	$1.04474\ 10^{11}$	28	7.42	62.81	70.49
5	$5.97615\ 10^{10}$	21	4.11	35.26	39.59
6	$3.59085\ 10^{10}$	50	3.91	83.50	87.61
7	$2.19832\ 10^{10}$	60	7.25	98.90	106.55
8	$1.33854\ 10^{10}$	45	3.59	73.00	76.86
9	$7.80442\ 10^{9}$	44	2.89	72.42	75.58
10	$6.44647\ 10^{9}$	50	2.14	81.92	84.33

TABLE 4
Fisher using weighted ACCPM.

M	Opt. sol.	# of iter.	AC time	Aux. time	Tot. time
2	152.347	19	14.70	18682.22	18697.59
3	78.8514	39	15.11	481.64	497.55
4	57.2284	51	8.87	495.76	505.49
5	46.4461	42	8.63	340.89	350.25
6	39.0399	68	9.05	574.05	584.04
7	34.2982	54	5.02	421.13	427.04
8	29.9889	88	9.98	684.29	695.37
9	27.7860	108	11.43	842.55	855.23
10	25.8340	81	5.12	622.63	628.92

and the CPU time spent in computation of the analytical center is very small.

It is worth stressing that computation times tend to diminish, or at least not to augment, with M. This is in sharp contrast with results for geometry-based branch-and-bound methods cited in the introduction.

While those test problems are larger than previously solved exactly, there are many larger ones for which one might want to obtain an exact solution. Of the five main ingredients of the proposed algorithm (heuristic VNS, column generation, AC-CPM, hyperbolic programming, and quadratic 0-1 programming) only one is the bottleneck, i.e., quadratic 0-1 programming. An efficient choice of weights in the ACCPM method accelerated considerably the algorithm for this step. Progress in solution methods for unconstrained quadratic 0-1 programming, a topic currently much studied, would immediately lead to exact solution of large instances of minimum sum-of-squares clustering.

The proposed algorithm could also be used in heuristic mode by canceling the quadratic 0-1 programming routine. As VNS often finds the optimal solutions of the subproblem this modified method might still often lead to an optimal solution, but without proof of its optimality.

Clearly, several of the new ideas proposed in this paper, i.e., exploitation of an initial solution to find bounds on dual variables, modification of columns instead of their removal when branching, and use of strategies to obtain small dual variables in the auxiliary problem, could apply to many other problems in clustering, location theory, and other fields.

REFERENCES

[1] J.P. BENZECRI, *Construction d'une classification ascendante hiérarchique par la recherche en chaîne des voisins réciproques*, Les Cahiers de l'Analyse des Données, VII (1982), pp. 209–218.

[2] M. DELATTRE AND P. HANSEN, *Bicriterion cluster analysis*, IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI, 2 (1980), pp. 277–291.

[3] G. DIEHR, *Evaluation of a branch and bound algorithm for clustering*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 268–284.

[4] W. DINKELBACH, *On nonlinear fractional programming*, Management Sci., 13 (1967), pp. 492–498.

[5] O. DU MERLE, *Points intérieurs et plans coupants: mise en œuvre et développement d'une méthode pour l'optimisation convexe et la programmation linéaire structurée de grande taille*, Ph.D. Thesis, HEC-Section of Management Studies, University of Geneva, Switzerland, 1995.

[6] O. DU MERLE, J.L. GOFFIN, AND J.P. VIAL, *On improvements to the analytic center cutting plane method*, Comput. Optim. Appl., 11 (1998), pp. 37–52.

[7] O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen, *Stabilized column generation*, Discrete Math., 194 (1999), pp. 229–237.
[8] A.W.F. Edwards and L.L. Cavalli-Sforza, *A method for cluster analysis*, Biometrics, 21 (1965), pp. 362–375.
[9] R.A. Fisher, *The use of multiple measurements in taxonomic problems*, Ann. Eugenics, VII part II (1936), pp. 179–188.
[10] P.C. Gilmore and R.E. Gomory, *A linear programming approach to the cutting stock problem*, Oper. Res., 9 (1961), pp. 849–859.
[11] F. Glover, *Tabu search—Part* I, ORSA J. Comput., 1 (1989), pp. 190–206.
[12] F. Glover, *Tabu search—Part* II, ORSA J. Comput., 2 (1990), pp. 4–32.
[13] J.-L. Goffin, A. Haurie, and J.-P. Vial, *Decomposition and nondifferentiable optimization with the projective algorithm*, Management Sci., 38 (1992), pp. 284–302.
[14] J. Gondzio, O. du Merle, R. Sarkissian, and J.-P. Vial, *ACCPM—A library for convex optimization based on an analytic center cutting plane method*, European Journal of Operational Research, 94 (1996), pp. 206–211.
[15] A.D. Gordon, *Classification: Methods for the Exploratory Analysis of Multivariate Data*, Chapman and Hall, New York, 1981.
[16] A.D. Gordon and J.T. Henderson, *An algorithm for Euclidean sum of squares classification*, Biometrics, 33 (1977), pp. 355–362.
[17] J.C. Gower and G.J.S. Ross, *Minimum spanning trees and single linkage cluster analysis*, Appl. Statistics, 18 (1969), pp. 54–64.
[18] P.L. Hammer and P. Hansen, *Logical relations in quadratic* 0-1 *programming*, Rev. Roumaine Math. Pures Appl., 26 (1981), pp. 421–429.
[19] P.L. Hammer, P. Hansen, and B. Simeone, *Roof duality, complementation and persistency in quadratic* 0–1 *optimization*, Math. Programming, 28 (1984), pp. 121–155.
[20] P. Hansen, B. Jaumard, and E. da Silva, *Average-linkage divisive hierarchical clustering*, J. Classification, to appear.
[21] P. Hansen, B. Jaumard, S. Krau, and O. du Merle, *A Column Generation Algorithm for the Weber Multisource Problem*, in preparation.
[22] P. Hansen, B. Jaumard, and N. Mladenović, *Minimum sum of squares clustering in a low dimensional space*, J. Classification, 15 (1998), pp. 37–55.
[23] P. Hansen, B. Jaumard, and B. Simeone, *Espaliers, a generalization of dendrograms*, J. Classification, 13 (1996), pp. 107–127.
[24] P. Hansen and N. Mladenović, *An introduction to variable neighborhood search*, in Metaheuristics. Advances and Trends in Local Search Paradigms for Optimization, S. Voss, S. Martello, I.M. Osman, and C. Roucairol, eds., Kluwer, Dordrect, The Netherlands, 1998, pp. 433–458.
[25] J.A. Hartigan, *Clustering Algorithms*, Wiley, New York, 1975.
[26] M. Inaba. N. Katoh, and H. Imai, *Applications of weighted Voronoi diagrams and randomization to variance-based k-clustering*, in Proceedings of the 10th ACM Symposium on Computational Geometry, Stony Brook, NY, 1994, pp. 332–339.
[27] S.C. Johnson, *Hierarchical clustering schemes*, Psychometrika, 32 (1967), pp. 241–245.
[28] L. Kaufman and P.J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, Wiley, New York, 1990.
[29] J.E. Kelley, *The cutting-plane method for solving convex programs*, J. SIAM, 8 (1960), pp. 703–712.
[30] S. Kirkpatrick, C.D. Gelatt, Jr., and M.P. Vecchi, *Optimization by simulated annealing*, Science, 20 (1983), pp. 671–680.
[31] W.L.G. Koontz, P.M. Narendra, and K. Fukunaga, *A branch and bound clustering algorithm*, IEEE Trans. Comput., C–24 (1975), pp. 908–915.
[32] G.N. Lance and W.T. Williams, *A general theory of classificatory sorting strategies.* 1. *Hierarchical systems*, The Computer J., 9 (1967), pp. 373–380.
[33] J.B. MacQueen, *Some methods for classification and analysis of multivariate observations*, in Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, 2, Berkeley, CA, 1967, pp. 281–297.
[34] N. Megiddo, *Combinatorial optimization with rational objective functions*, Math. Oper. Res., 4 (1979), pp. 414–424.
[35] M. Minoux, *Programmation mathématique: théorie et algorithmes*, Tome 2, Dunod (Bordas), Paris, 1983.
[36] B. Mirkin, *Mathematical Classification and Clustering*, Kluwer, Dordrecht, The Netherlands, 1996.

[37] N. Mladenović and P. Hansen, *Variable neighborhood search*, Comput. Oper. Res., 24 (1997), pp. 1097–1100.
[38] F. Murtagh, *A survey of recent advances in hierarchical clustering algorithms*, Comput. J., 26 (1983), pp. 329–340.
[39] M.R. Rao, *Cluster analysis and mathematical programming*, J. Amer. Statist. Assoc., 66 (1971), pp. 622–626.
[40] E.H. Ruspini, *Numerical methods for fuzzy clustering*, Inform. Sciences, 2 (1970), pp. 319–350.
[41] D.M. Ryan and B.A. Foster, *An integer programming approach to scheduling*, in Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling, A. Wren, ed., North-Holland, Amsterdam, 1981, pp. 269–280.
[42] S. Schaible, *Fractional programming*, in Handbook of Global Optimization, R. Horst and P.M. Pardalos, eds., Kluwer, Dordrecht, The Netherlands, 1995, pp. 495–608.
[43] H. Späth, *Cluster Analysis Algorithms for Data Reduction and Classification of Objects*, Ellis Horwood, Chichester, UK, 1980.
[44] H.D. Vinod, *Integer programming and the theory of grouping*, J. Amer. Statist. Assoc., 64 (1969), pp. 506–519.
[45] J.H. Ward, Jr., *Hierarchical grouping to optimize an objective function*, J. Amer. Statist. Assoc., 58 (1963), pp. 236–244.