Journal of Global Optimization (2005) 32:281–306
DOI 10.1007/s10898-004-2706-7

A Global Optimization RLT-based Approach for Solving the Hard Clustering Problem

HANIF D. SHERALI and JITAMITRA DESAI
Grado Department of Industrial and Systems Engineering (0118), Virginia Polytechnic Institute and State University, 250 Durham Hall, Blacksburg, VA 24061, U.S.A. (e-mail: hanifs@vt.edu, jidesai@vt.edu)

(Received 1 April 2004; accepted in revised form 7 April 2004)

Abstract. The field of cluster analysis is primarily concerned with the sorting of data points into different clusters so as to optimize a certain criterion. Rapid advances in technology have made it possible to address clustering problems via optimization theory. In this paper, we present a global optimization algorithm to solve the *hard* clustering problem, where each data point is to be assigned to exactly one cluster. The hard clustering problem is formulated as a nonlinear program, for which a tight linear programming relaxation is constructed via the Reformulation-Linearization Technique (RLT) in concert with additional valid inequalities that serve to defeat the inherent symmetry in the problem. This construct is embedded within a specialized branch-and-bound algorithm to solve the problem to global optimality. Pertinent implementation issues that can enhance the efficiency of the branch-and-bound algorithm are also discussed. Computational experience is reported using several standard data sets found in the literature as well as using synthetically generated larger problem instances. The results validate the robustness of the proposed algorithmic procedure and exhibit its dominance over the popular *k*-means clustering technique. Finally, a heuristic procedure to obtain a good quality solution at a relative ease of computational effort is also described.

Key words: Clustering problem, Global optimization, Hard clustering, *k*-Means algorithm, Reformulation-Linearization Technique

1. Introduction

In many applications, data is generated that needs to be analyzed and deciphered in order to extract patterns or information from it. One approach to sift this data is to solve the underlying *clustering problem*. In a broad sense, this involves the process of partitioning the given data set into subsets called *clusters*, such that some accumulated distance measure between points belonging to common clusters is minimized. Several clustering approaches have been developed to effectively analyze and interpret large volumes of data information. Such clustering problems arise in a wide scope of applications related to cellular manufacturing, medicine, archaeology, and marketing (see Hartigan (1975) for a detailed survey on applications of cluster analysis).

Mangiameli et al. (1996) have shown that the clustering problem is NP-Hard and thus, finding a global optimum to this problem is a computationally onerous task. However, a significant reduction in computational effort can be achieved by considering judiciously defined subsets of the original data set and applying a more refined partitioning scheme, separately to each such subset, to arrive at the final clustering pattern. This concept has lead to a variety of clustering approaches such as statistical methods, self-organizing maps, hierarchical clustering, and a limited number of optimization techniques.

The most popular among these methods is the hierarchical clustering method. Hierarchical clustering of data sets can be achieved by two types of splitting methods: partitive splitting and agglomerative merging. The hierarchical clustering technique that is most widely in vogue is the agglomerative approach (see Ward, 1963, Sultan et al., 2002). This begins with individual data points being singleton clusters, and then at successive iterations, merges them to generate a tree structure. This tree is referred to as a *dendrogram*. The dendrogram is cut off at some level at which a large distance is observed between pairs of clusters. This approach does not usually provide a unique clustering, and in fact, does not guarantee that intra-cluster distance is minimized. To obtain an optimal clustering using this approach, the dendrogram must be subdivided at several points (Sultan et al., 2002).

In contrast with this method, partitive clustering initially divides the data set into a predefined number of clusters by minimizing some criterion (usually a distance measure). Then, at each iteration, the intra-cluster distance is minimized and the inter-cluster distance is maximized (Sultan et al., 2002). In general, solution techniques based on hierarchical clustering have problems related to robustness and uniqueness of the solution obtained (Lukashin and Fuchs, 2000). On the other hand, the limited number of optimization techniques that are available cannot guarantee that the derived solution is a global optimum. Moreover, while the number of clusters used is typically prescribed as a fixed, external parameter for the algorithm being utilized, there is some interest in also simultaneously determining an optimal number of clusters to use, perhaps given a fixed cost associated with constructing each cluster (see Dubes, 1987; Jung et al., 2003).

Another important factor involved in solving clustering problems is the distance measure under consideration. Obviously, optimal clustering depends on the distance measure being used. Distance measures are divided into *metric* and *semi-metric* measures (Sultan et al., 2002), and most hierarchical procedures (that are based on the nearest neighbor approach) utilize either one of these measures. A semi-metric distance measure is one that satisfies the following properties for any two vectors i and j in a given data set: (1) the distance between i and j is positive, i.e., $d_{ij} > 0$; (2) $d_{ij} = d_{ji}$,

and (3) $d_{ii} = 0$. In addition to the above properties, if a distance measure satisfies the triangle inequality, i.e., $d_{ij} + d_{jk} \geqslant d_{ik}$, then it qualifies as a metric measure.

In this paper, we consider the *hard clustering* problem wherein each data point must be assigned to *exactly* one cluster. This is in contrast to other clustering problems where a data point may belong to several clusters with a membership grade assigned to each data point that represents the likelihood of the data point belonging to that cluster. Such a problem is referred to as a *fuzzy clustering* problem (the word fuzzy is derived from fuzzy programming, and reflects the fact that the specific cluster to which a data point belongs is only fuzzily identified, and is not described deterministically).

The hard clustering problem has been extensively dealt with in the literature and there are several approaches that have been explored to solve this problem. The first attempt to solve the clustering problem was by using the k-means algorithm (Forgy, 1966; McQueen, 1967). This method is widely used in practice, but often fails to produce a global optimum. Several optimization techniques such as dynamic programming (Jensen, 1969), convexity cuts (Selim, 1982), alternative cutting plane algorithms (Groetschel and Wakabayashi, 1989), lagrangian relaxation methods (Mulvey and Crowder, 1979) and integer programming formulations coupled with branch-and-bound strategies (Vinod, 1969; Rao, 1971; Koontz et al., 1975) have been used to solve the hard clustering problem. Of recent flavor are meta-heuristic search methods such as simulated annealing, tabu search, and the genetic algorithm. Klein and Dubes (1989) and Selim and Al-Sultan (1991) were the first to study a simulated annealing approach in this context, and thenceforth, several other modifications of this procedure have been proposed. Al-Sultan (1995) developed a tabu search algorithm, and Bhuyan et al. (1991) and Krovi (1992) have advocated a framework using the genetic algorithm to solve the hard clustering problem. Computational experience along with a comparison between four heuristic algorithms that solve the hard clustering problem has been provided by Al-Sultan and Khan (1996).

In this research effort, we design an optimization approach based on the *Reformulation-Linearization Technique* (RLT) (refer Sherali and Adams, 1990, 1994, 1999; Sherali and Tuncbilek, 1992, 1995) to solve the hard clustering problem. The underlying nonlinear, discrete optimization problem is transformed into an equivalent 0–1 mixed-integer program having a tight linear programming (LP) relaxation as prescribed by the RLT, and a specialized algorithm is designed to derive a global optimum.

The remainder of this paper is organized as follows. Section 2 provides a series of enhanced formulations of the problem based on RLT constructs as well as the derivation of certain classes of valid inequalities. Accordingly, a

tailored branch-and-bound global optimization algorithm is also delineated in Section 2. Section 3 presents computational results using certain standard test problems from the literature as well as using larger synthetically generated data sets, and explores the performance of different formulations and implementation strategies. Finally, Section 4 concludes the paper with a summary and a discussion on further avenues for research in this area.

2. Hard Clustering Problem

The hard clustering problem can be defined as follows. Given a set of n data points, each having some s attributes, we are required to assign each of these points to exactly one of some c clusters (where c is given), so as to minimize the total squared Euclidean distance between the data points and the centroid of the clusters to which they are assigned. That is to say, if data point i, having a location descriptor $a_i \in \mathbb{R}^s$ is assigned to cluster j having a to-be-determined centroid $z_j \in \mathbb{R}^s$, then the associated penalty is assumed to be proportional to the square of the straight line distance separation between a_i and z_j in $\mathbb{R}^s$. An optimal solution to the clustering problem determines the cluster configuration such that the sum of all such distances is minimized. This problem can be mathematically stated as follows.

$$\textbf{CP1 :} \quad \text{Minimize} \ \sum_{i=1}^{n} \sum_{j=1}^{c} w_{ij} \left\| a_i - z_j \right\|^2 \tag{1a}$$

$$\text{subject to} \ \sum_{j=1}^{c} w_{ij} = 1, \quad \forall i = 1, \ldots, n \tag{1b}$$

$$w \geqslant 0, \tag{1c}$$

where $a_i = (a_{ik}, k = 1, \ldots, s)^{\mathrm{T}}$, and $z_j = (z_{jk}, k = 1, \ldots, s)^{\mathrm{T}}$, and the norm $\|\|$ in (1a) represents the Euclidean distance between the two points in its argument in the s-dimensional space under consideration. We assume that $n > c$, because otherwise, the problem would be trivially solved by simply designating each point to constitute a cluster by itself. Observe also that for any fixed z, w will automatically be binary-valued at a resultant extreme point optimum.

For a fixed w, optimality of the resulting convex objective function in z requires that

$$\sum_{i=1}^{n} w_{ij}(z_{jk} - a_{ik}) = 0, \quad \forall j, k, \tag{2a}$$

that is,

$$z_{jk} = \frac{\sum_{i=1}^{n} w_{ij} a_{ik}}{\sum_{i=1}^{n} w_{ij}}, \quad \forall j, k, \tag{2b}$$

where the denominator in (2b) is positive at optimality under our assumption that $n > c$.

Consequently under the conditions (2a) and (2b), we have that the objective function (1a) is equivalently given by

$$\begin{aligned}
&\sum_{i=1}^{n}\sum_{j=1}^{c}\sum_{k=1}^{s} w_{ij}(z_{jk} - a_{ik})^2 \\
&\quad = \sum_{i=1}^{n}\sum_{j=1}^{c}\sum_{k=1}^{s} w_{ij}(z_{jk} - a_{ik}) z_{jk} - \sum_{i=1}^{n}\sum_{j=1}^{c}\sum_{k=1}^{s} w_{ij}(z_{jk} - a_{ik}) a_{ik} \\
&\quad = \sum_{i=1}^{n}\sum_{j=1}^{c}\sum_{k=1}^{s} w_{ij} a_{ik}^2 - \sum_{i=1}^{n}\sum_{j=1}^{c}\sum_{k=1}^{s} a_{ik} w_{ij} z_{jk}.
\end{aligned} \tag{3}$$

By (1b), noting that $\sum_{i=1}^{n}\sum_{j=1}^{c}\sum_{k=1}^{s} w_{ij} a_{ik}^2 = \sum_{i=1}^{n}\sum_{k=1}^{s} a_{ik}^2$, a constant, we have that CP1 can be equivalently solved via the following problem.

$$\textbf{CP1.1 :} \quad \text{Maximize} \sum_{i=1}^{n}\sum_{j=1}^{c}\sum_{k=1}^{s} a_{ik} w_{ij} z_{jk} \tag{4a}$$

$$\text{subject to } z_{jk}\sum_{i=1}^{n} w_{ij} - \sum_{i=1}^{n} a_{ik} w_{ij} = 0, \quad \forall j, k \tag{4b}$$

$$\sum_{j=1}^{c} w_{ij} = 1, \quad \forall i = 1, \ldots, n \tag{4c}$$

$$w \text{ binary}, \tag{4d}$$

where (4d) has been explicitly imposed to exploit this optimality condition in the algorithmic process.

Note that by (4b), if we denote for any given solution w to CP1.1, the sets

$$S_j = \{i : w_{ij} = 1\}, \quad \forall j, \tag{5}$$

then we have

$$z_{jk} = \sum_{i \in S_j} a_{ik} \Big/ |S_j|, \quad \forall k, \text{ for each } j, \tag{6}$$

or that the vector z_j is a convex combination (with equal weights) of the points a_i, $i \in S_j$. Let us now define $I_j \subseteq \{1, \ldots, n\}$ as the set of potential points $i \in \{1, \ldots, n\}$ that are assignable to cluster j (in the absence of any relevant information or algorithmic restrictions, we would have $I_j \equiv \{1, \ldots, n\}), \forall j = 1, \ldots, c$, and let us denote

$$H(I_j) = \text{conv}\{a_i : i \in I_j\}$$
$$\subseteq \left\{ z_j : \sum_{k=1}^{s} \gamma^j_{qk} z_{jk} \leqslant \gamma^j_{q0} \text{ for } q = 1, \ldots, Q_j \right\} \equiv \bar{H}(I_j), \text{say}, \ \forall j, \quad (7a)$$

where the set of Q_j inequalities in (7a) defines some bounded superset $\bar{H}(I_j)$ of $H(I_j)$. Observe that for notational convenience, we have used the superscript j in lieu of I_j for the inequalities describing $\bar{H}(I_j)$ in (7a), and also, note that for isomorphic subsets of $\{1, \ldots, n\}$, we can use the same description of $\bar{H}(\cdot)$. In the simplest case, $\bar{H}(I_j)$ might be taken as an enclosing hyperrectangle as expounded below. Note that $H(I_j)$ is efficiently computable in polynomial time for points in two-dimensions using the method described in Manber (1989). (For example, the Graham's scan algorithm produces the convex hull in $O(n \log n)$ steps.) However, for higher dimensions, computing the convex hull can prove to be an expensive task. Nevertheless, under some specific assumptions, it has been shown in the literature that the convex hull can be obtained for higher dimensions using techniques such as neural networks (refer Leung et al., 1997), cutting planes (Chazelle, 1991), and direct convex hull computations for convex polyhedra (refer Klapper, 1987; Balas, 1988). In the context of our problem, we can gainfully employ any such technique to derive suitable valid inequalities for constructing $\bar{H}(I_j), \forall j$. For simplicity, regardless of problem dimension, we will take $\bar{H}(I_j)$ to be a hyperrectangle that bounds the collection of points $a_i, i \in I_j$, as defined below for each j.

$$\bar{H}(I_j) = \left\{ z_j : \alpha^j_k \leqslant z_{jk} \leqslant \beta^j_k, k = 1, \ldots, s \right\}, \quad (7b)$$

where,

$$\alpha^j_k = \min\{a_{ik} : i \in I_j\}, \quad \forall k, \text{ and } \beta^j_k = \max\{a_{ik} : i \in I_j\}, \quad \forall k, \text{ for each } j. \quad (7c)$$

Additionally, we could incorporate other valid inequalities that are valid for $H(I_j)$ within (7b). In order to maintain generality in presentation of these various viable algorithmic strategies, we will henceforth assume that some such suitable set $\bar{H}(I_j)$ as given by (7a) has been obtained.

Now, we can impose the implied constraints defining $\bar{H}(I_j)$ for each j within CP1.1, where, prior to any further analysis, $I_j \equiv \{1, \ldots, n\}, \forall j$. (Subsequently, we will be modifying the sets I_j iteratively in a branch-and-bound context.) However, instead of simply imposing these constraints, let us impose the product of these constraints with each w_{ij} and $(1 - w_{ij})$, $\forall i \in I_j$, for each $j = 1, \ldots, n$, in the spirit of RLT. This yields the following restatement of CP1.1.

CP1.2 : Maximize $\sum_{i=1}^{n} \sum_{j=1}^{c} \sum_{k=1}^{s} a_{ik} w_{ij} z_{jk}$ (8a)

$$\text{subject to } z_{jk}\sum_{i=1}^{n} w_{ij} - \sum_{i=1}^{n} a_{ik}w_{ij} = 0, \quad \forall j,k \tag{8b}$$

$$\sum_{k=1}^{s} \gamma_{qk}^{j} z_{jk} w_{ij} \leqslant \gamma_{q0}^{j} w_{ij}, \quad \forall i \in I_j, \forall j, \forall q \tag{8c}$$

$$\sum_{k=1}^{s} \gamma_{qk}^{j} \left(z_{jk} - z_{jk} w_{ij}\right) \leqslant \gamma_{q0}^{j} \left(1 - w_{ij}\right), \quad \forall i \in I_j, \forall j, \forall q \tag{8d}$$

$$\sum_{j=1}^{c} w_{ij} = 1, \quad \forall i = 1, \ldots, n \tag{8e}$$

$$w \in W, \tag{8f}$$

where,

$$W = \left\{w \text{ binary: } w_{ij} = 0 \text{ for all}(i,j) \in I^{-}, w_{ij} = 1 \text{ for all}(i,j) \in I^{+}\right\} \tag{9}$$

and where

$$I^{+} = \left\{(i,j) : w_{ij} \text{ has been fixed at 1 (subject to (8e)}\right\}, \tag{10a}$$

$$I^{-} = \left\{(i,j) : w_{ij} \text{ has been fixed at 0 }\right\}, \tag{10b}$$

$$I^{f} = \left\{(i,j) : w_{ij} \text{ is free (i.e., not fixed)}\right\}. \tag{10c}$$

Note that,

$$I_j = \left\{i \in \{1, \ldots, n\} : (i,j) \in I^{+} \cup I^{f}\right\} \equiv \{i \in \{1, \ldots, n\} : (i,j) \notin I^{-}\}. \tag{11a}$$

Also, for each $i \in \{1, \ldots, n\}$, define $J_i \subseteq \{1, \ldots, c\}$ as the set of assignable clusters for data point i, i.e.,

$$J_i \equiv \left\{j \in \{1, \ldots, c\} : (i,j) \in I^{+} \cup \mathrm{I}^{f}\right\} \equiv \{j \in \{1, \ldots, c\} : (i,\ j) \notin I^{-}\}. \tag{11b}$$

Hence, whenever $I^{-} = \varnothing$ (*e.g.* to initialize the algorithm), we have $J_i \equiv \{1, \ldots, c\}, \forall i = 1, \ldots, n$.

There are two other classes of constraints that we can add to (8a)–(8f) in order to tighten its representation. The first is based on the valid restrictions

$$1 \leqslant \sum_{i=1}^{n} w_{ij} \leqslant n - c + 1, \quad \forall j = 1, \ldots, c, \tag{12a}$$

which asserts that each cluster should be assigned at least one point, and so, each cluster contains at most $n-c+1$ points due to hard clustering. Furthermore, constraints (12a) can also be multiplied by $z_{jk}, \forall k$, for each j, in order to generate the following RLT constraints

$$z_{jk} \leqslant \sum_{i=1}^{n} w_{ij} z_{jk} \leqslant (n-c+1) z_{jk}, \quad \forall (j,k). \tag{12b}$$

REMARK 1. Note that the right-hand inequalities in (12a) itself, being implied by (8e, f) and the left-hand inequalities in (12a), can be omitted. While the right-hand inequalities in (12b) might be useful, their worth is questionable. Hence, these inequalities, as well as the utility of other RLT constraints (including those in (12a, b)) and related modeling strategies were empirically investigated to ascertain their merit, before proposing a final model. Computational results indicated that these inequalities did at least marginally improve the algorithmic convergence (refer Table 5 for relevant results). Let us refer to CP1.2 enhanced by the additional valid inequalities (12a, b) as CP1.3.

REMARK 2. A strong factor that can potentially weaken the relaxation of CP1.2 and contribute towards its difficulty in solving via a branch-and-bound approach is the *symmetry* in the problem structure. Note that for any given solution, alternative equivalent solutions could be obtained by simply re-indexing each cluster composition. To circumvent this difficulty, we propose two alternative sets of hierarchical constraints that could be used to defeat the symmetry (see Sherali and Smith, 2001, for a general discussion on this subject).

SYMMETRY STRATEGY 1.

Impose the following constraints:

$$w_{11} = 1, w_{1j} = 0, \quad \forall j = 2, \ldots, c \tag{13a}$$

$$\sum_{i=1}^{n} w_{ij} \geq \sum_{i=1}^{n} w_{i,j+1}, \quad \forall j = 2, \ldots, c-1. \tag{13b}$$

Note that we can arbitrarily assign some point, say point $i = 1$ as in (13a), to the first cluster. For the remaining clusters, to impart some distinctive identity to these sets, we can require that the indexing be performed in nonincreasing order of their size. This is represented by (13b). Of course, whenever a solution includes clusters having common sizes, we could still produce alternative equivalent solutions by re-indexing. However, (13b) does curtail this phenomenon.

SYMMETRY STRATEGY 2.

Initialization: Put the counter $r = 1$. Find a point $p_1 \in \underset{i=1,\ldots,n}{\text{arglexmin}}\{a_{i1}, a_{i2}, \ldots, a_{is}\}$.

Step 1. If $r = c-1$, go to Step 3. Else proceed to Step 2.

Step 2. Find a point $p_{r+1} \in \underset{\substack{i=1,\ldots,n \\ i \neq p_1,\ldots,p_r}}{\text{argmax}} \left\{ \underset{t=1,\ldots,r}{\text{minimum}} \left\| a_i - a_{p_t} \right\|^2 \right\}$.

Increment r by 1, and return to Step 1.

Step 3. Impose the constraints

$$\left\{ w_{p_r j} = 0 \text{ for } j = r+1, \ldots, c \right\} \text{for each } r = 1, \ldots, c-1 \tag{14}$$

Note that the assertion (14) is valid because we can restrict each of the identified points p_r to belong to one of the first r clusters, for each $r = 1,\ldots, c-1$. By having selected a dispersed set of points following the process in Step 2 above (given the "corner" point p_1 selected at the initialization step), we enhance the likelihood that these points also turn out to belong to different clusters, thereby imparting a specific identity to each cluster. As before, this tends to eliminate the symmetry effect, although not completely. In our computational experiments, we test the relative merits of these two symmetry-defeating strategies.

The augmented problem CP1.2 using (12a, b) along with (13a, b) or (14) can be restated as follows, where we have substituted

$$y_{ijk} = w_{ij} z_{jk}, \quad \forall i, j, k \tag{15}$$

in the spirit of RLT, recognizing Proposition 1 as given below.

CP1.4:

$$\text{Maximize} \sum_{i \in I_j} \sum_{j \in J_i} \sum_{k=1}^{s} a_{ik}\ y_{ijk} \tag{16a}$$

$$\text{subject to} \quad \sum_{i \in I_j} y_{ijk} - \sum_{i \in I_j} a_{ik} w_{ij} = 0, \quad \forall j, k \tag{16b}$$

$$\sum_{k=1}^{s} \gamma^j_{qk} y_{ijk} \leqslant \gamma^j_{q0} w_{ij}, \quad \forall i \in I_j, \forall j, \forall q \tag{16c}$$

$$\sum_{k=1}^{s} \gamma^j_{qk} \left(z_{jk} - y_{ijk} \right) \leqslant \gamma^j_{q0} \left(1 - w_{ij} \right), \quad \forall i \in I_j, \quad \forall j, \forall q \tag{16d}$$

$$\sum_{j \in J_i} w_{ij} = 1, \quad \forall i = 1, \ldots, n \tag{16e}$$

$$\sum_{i \in I_j} w_{ij} \geq 1, \quad \forall j = 1, \ldots, c \tag{16f}$$

$$z_{jk} \leqslant \sum_{i \in I_j} y_{ijk} \leq (n - c + 1) z_{jk}, \quad \forall j, k \tag{16g}$$

$$\text{Constraints(13a, b) or (14)} \tag{16h}$$

$$w \in W. \tag{16i}$$

PROPOSITION 1. *For any feasible solution to* (16a) – (16i)*, we have that* (15) *holds true. Hence,* (16a)–(16i) *is an equivalent linear* 0-1 *mixed integer programming* (MIP) *representation of CP1.*

Proof. For any (i, j), suppose that $w_{ij} = 0$. Since$\bar{H}(I_j)$ (as defined in (7a, b)) is a bounded set, its homogeneous system has a unique solution given by the 0-vector. Hence, by (16c), we have that $y_{ijk} \equiv 0, \forall k$, and therefore, (15) holds true in this case. Similarly, if $w_{ij} = 1$, for any (i, j), then (16d) implies that $(z_{jk}–y_{ijk}) = 0, \forall k$, or that (15) again holds true. This completes the proof.

We can now design a branch-and-bound algorithm to solve CP1.4 based on the following specialized features, as opposed to using default strategies of a standard MIP solver such as CPLEX-MIP 8.1.0 for this purpose.

(a) Upper bounds can be computed by using the LP relaxation to (16a)–(16i). Note that in formulating (16a)–(16i), given (10a)–(10c), for any partial solution corresponding to a node subproblem, we redefine I_j, J_i, and $\bar{H}(I_j)$ as in (11a), (11b), and (7a)–(7c) respectively, and use this to reconstruct the model representation, including the derivation of (16c) and (16d).

(b) Heuristic solutions can be derived at each node based on a rounding scheme applied to the LP solution. Specifically, denoting $\bar{w}$ as part of the LP relaxation solution obtained for any node subproblem, if $\bar{w}$ is binary-valued, then by Proposition 1, the LP solution is optimal for the node subproblem and directly provides a feasible solution for CP1.4 (as well as CP1). We can therefore fathom this node and update the incumbent solution, if necessary. Otherwise, we can round the $\bar{w}$ solution to the nearest binary solution subject to (16e) (also subsequently ensuring (16f), i.e., each cluster inherits at least one assignable point). Here, for each data point u, we determine $\bar{w}_{uv} = \max\{\bar{w}_{uj} : j \in J_u\}$, with ties broken by selecting a cluster having the smallest value for $\left|I_j\right|$ and we assign $\bar{w}_{uv} = 1$ and $\bar{w}_{uj} = 0, \forall j \in J_u$,

$j \neq v$. (A more comprehensive tie-breaking rule would be to evaluate the objective function (given by (1a)) corresponding to all possible alternative rounded solutions, and pick the best one among them. However, this would lead to a considerably greater computational effort at each node, and was therefore not implemented in our computations.) Using this resulting binary $\bar{w}$ solution, we then compute the corresponding z-values using (2b), and hence obtain a feasible solution for CP1, which can be used to possibly update the incumbent solution for CP1.4, upon invoking (15). (For large-scale problems, the overall procedure could be terminated after applying such an LP based heuristic method at node-zero itself, or by using some limited branching scheme in order to prescribe a heuristic solution to the problem.)

(c) To select a branching variable, we compute the total absolute discrepancy in the linearized objective terms in (16a) relative to the nonlinear product terms these represent according to (15), as given by

$$\theta_{ij} = \sum_{k=1}^{s} \left| a_{ik}(\bar{y}_{ijk} - \bar{w}_{ij}\bar{z}_{jk}) \right|, \quad \forall (i,j) \tag{17a}$$

where $(\bar{w}, \bar{z}, \bar{y})$ solves the LP relaxation $\overline{\textbf{CP1.4}}$ to CP1.4 given by (16a) – (16i). Then, in one partitioning strategy, we branch on the dichotomy that $w_{uv} = 0$ or 1, where

$$(u, v) \in \arg\max_{(i,j) \in I^f} \{\theta_{ij}\}. \tag{17b}$$

Naturally, on the branch $w_{uv} = 1$, we also set $w_{uj} = 0$, $\forall j \neq v$, and on the branch $w_{uv} = 0$, the sets I_v and J_u would now not include the respective indices u and v, and $\bar{H}(I_v)$ would accordingly exclude a_u in the convex hull computation or its approximation. □

REMARK 3. Exploiting the structure of the inherent generalized upper bounding (GUB) constraints (16e), we also explore an alternative *specially ordered set* (SOS) branching strategy. In this scheme, defining θ_{ij} as in (17a), and denoting $\theta_i \equiv \sum_{j \in J_i} \theta_{ij}$, we compute $u \in \arg\max\{\theta_i\}$. (Note that by Proposition 1, if $\theta_u > 0$ then the vector $(\bar{w}_{uj},\ j \in J_u)$ is not binary-valued; else, we simply select u such that the total fractionality of the components of this latter vector is a maximum.) We now partition J_u into two children nonempty sets, J_{u1} and J_{u2}, as follows, where we then construct two subproblem nodes in the branch-and-bound tree corresponding to the respective imposed branching restrictions $\sum_{j \in J_{u1}} w_{uj} = 1$ and $\sum j \in J_{u2}\, w_{uj} = 1$. To determine this partition J_{u1} and J_{u2} of J_u, we first arrange the θ_{uj} values, $j \in J_u$, in nonincreasing order. Let this sorted set be $\{\theta_{uj_1}, \theta_{uj_2}, \ldots, \theta_{uj_l}\}$, where $l = |J_u| \geq 2$. Now, define $p \geq 1$ to be the smallest

integer such that $\sum_{r=1}^{p} \theta_{uj_r} \geq \theta_u/2$. Note that $1 \leqslant p < l$, by virtue of the sorted list. Accordingly, we then define $J_{u1} = \{j_1, \ldots, j_p\}$and $J_{u2} = \{j_{p+1}, \ldots, j_l\}$. Likewise, for each of these children nodes, the sets I_j would then be revised accordingly. (d) Using the LP dual solution, a reduced cost cut based on requiring the objective function to be greater than or equal to the incumbent lower bound can be constructed in terms of w, by surrogating and dualizing all constraints except for (16e) and $0 \leqslant w_{ij} \leqslant 1, \forall (i,j)$. Logical tests can be conducted on this in order to possibly fix some w-variables at 0 or 1 values, and thereby tighten the relaxation further (at least for the children nodes, if not for resolving the LP at the same node).

REMARK 4. In our implementation of the branch-and-bound algorithm that includes features (a)–(d) outlined above, a depth-first strategy was adopted to develop the enumeration tree. For the purpose of obtaining tight lower bounds, and to possibly update incumbent solutions, a rounding heuristic as proposed in (b) was employed at every node of the branch-and-bound tree. Also, based on our computational experiments, the SOS branching was determined to be the best branching strategy for larger problem instances (refer Table 6 for pertinent results), and was therefore taken as the primary branching scheme. Here, in the depth-first framework, we branched first along the J_{u1} side to explore the corresponding child node. (Note that for problems having a large number of clusters, to find good feasible solutions more quickly, we could first explore the child node along the side having the smaller $|J_{u1}|$ or $|J_{u2}|$ value, breaking ties by choosing J_{u1}). The overall branch-and-bound algorithm was implemented in C++, and the commercial software CPLEX 8.1.0 was invoked for the purpose of solving the LP relaxations at each node. Furthermore, the optimal basis for the parent node was used as an advanced-start basis for the two children nodes, thereby enabling a quicker update for the solutions to each of the node subproblems. Note that the CPLEX 8.1.0 command options facilitate these implementations.

3. Computational Results

Throughout this section, we will use the following terminology:

- UB_0: Optimal objective function value of $\overline{\text{CP1.4}}$ at node zero.
- LB_0: Objective function value of the heuristic solution to CP1.4 found at node zero.
- Z_0^*: Objective function value of CP1 corresponding to the heuristic solution found at node zero.
- LB*: Optimal objective function value of CP1.4.

Z^*: Optimal objective function value of CP1, evaluated at the optimal solution to CP1.4.

$Z^*_{k-\text{means}}$: Best objective function value obtained via the k-means algorithm.

CPU*: CPU time required to determine a global optimum for CP1.4 via the proposed branch-and-bound algorithm.

CPU$_{k-\text{means}}$: CPU time required for the k-means algorithm.

CPU$_0$: CPU time required to determine a heuristic solution at node zero via the solution to $\overline{\text{CP1.4}}$.

First, for the purpose of illustration, consider the following clustering problem having ten data points to be divided into three clusters, where each data point is assigned two attributes, (i.e., $s = 2$). Table 1 provides the input data, for this example problem. Using the above data, the LP relaxation of Problem CP1.4 was solved and the optimal solution value at node zero (UB_0) was found to be 43156. Applying the rounding heuristic described in Section 2, we obtained an incumbent value (LB_0) of 22434.68. Hence, the gap ratio at node zero is given by $UB_0/LB_0 = 1.9236$. Actually, since LB$^* = 27884.5$, as determined below, the true LP-IP gap at node zero is $UB_0/LB^* = 1.5476$. Also, the objective function value for the minimization problem CP1 (computed by substituting the (w, z) parts of the heuristically determined node zero incumbent solution into (1a)) was found to be $Z^*_0 = 18484.35$. Next, using the SOS branching strategy designed in (17a) and Remark 3, from the LP solution at node zero we get $\theta_3 = \arg\max\{\theta_i\} = 1174$, and the corresponding θ_{3j} values are given by $\{766, 300.44, 107.56\}$. Hence, $p = 1$, and we can now formulate the subnode problems by splitting $J_3 \equiv \{1, 2, 3\}$ into two subsets, $J_{3,1} = \{1\}$ and $J_{3,2} = \{3, 2\}$, and respectively imposing the constraints $w_{31} = 1$ and $w_{32} + w_{33} = 1$ for the corresponding subnode problems. Employing a depth-first strategy, Figure 1 depicts the (partial) branch-and-bound tree generated up to node eight, and illustrates the SOS branching computations. Continuing to completion, the optimal objective function value for Problem CP1.4 was found to be $LB^* = 27884.5$, and the corresponding $Z^* = 15805.25$. A total of 27 nodes were enumerated in determining this optimal solution. For the purpose of comparison, the above problem was solved via the k-means algorithm, and the optimal objective function value for Problem CP1 was found to be $Z^*_{k-\text{means}} = 34404.857$. (Note that, since the k-means algorithm requires the cluster centers as an input, its performance can be significantly enhanced by a

Table 1. Attributes for the 10 data points in R^2 for the illustrative example

Attributes	Points									
	1	2	3	4	5	6	7	8	9	10
1	−57	54	46	8	−36	−22	34	74	−6	21
2	28	−65	79	111	52	−76	129	6	−41	45

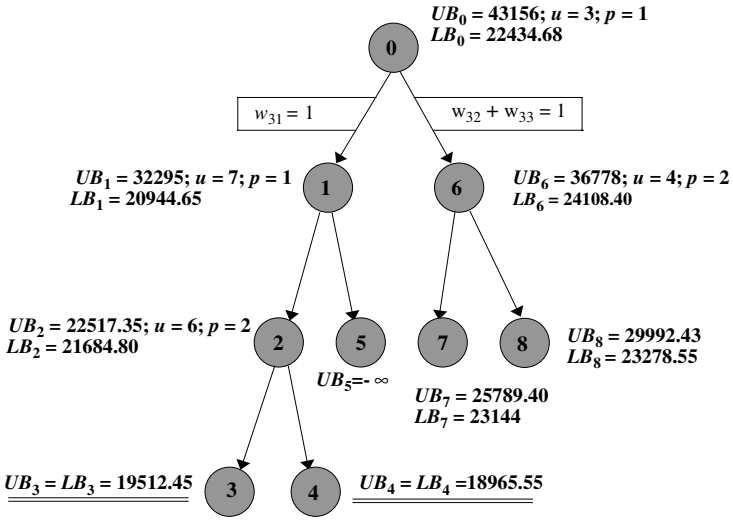
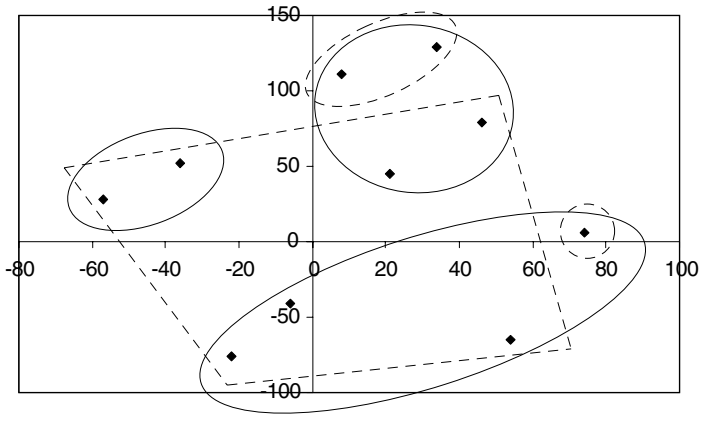


Figure 1. Branch-and-bound tree illustrating the SOS branching strategy.

good estimate of the initial cluster centers. In our computations, five randomly generated cluster center configurations were examined, and the best resulting solution was used for the above comparison). Considering the ratio $Z^*_{k-\text{means}}/Z^* = 2.176$, it is evident that the performance of the optimization algorithm is considerably superior to the k-means algorithm. Indeed, even from the ratio $Z^*_{k\text{-means}}/\ Z^*_0 = 1.86$, we see that the feasible solution obtained from node zero of the optimization problem itself is significantly better than the k-means solution. Figure 2 displays the optimal clustering patterns obtained via the proposed optimization algorithm (solid lines) and via the k-means algorithm (dashed lines).



Solid line: Optimal clustering; Dashed line: k-means clustering.

Figure 2. Clustering patterns obtained by solving Problem CP1.4 via the proposed algorithm and by the k-means algorithm.

Next, we used the following standard data sets given in Späth (1980) to test our proposed methodology:

1. *Data Set 1*. This is a set of Cartesian coordinates for 22 German towns, which yields a clustering problem having 22 points in a two-dimensional space.
2. *Data Set 2*. This is a set of Cartesian coordinates for 59 German towns, which yields a clustering problem having 59 points in a two-dimensional space.
3. *Data Set 3*. This pertains to 89 postal zones in Germany, where each zone has three attributes, namely, surface area (measured in square kilometers), population, and the density of population. This yields a clustering problem having 89 points in a three-dimensional space.
4. *Data Set 4*. This is also based on the 89 postal zones of Data Set 3, but considers four attributes, namely, the number of self-employed people, civil servants, clerks, and manual workers. This yields a clustering problem having 89 points in a four-dimensional space.

The above-mentioned data sets were used to provide the input data for Problem CP1.4, and the performance of the proposed approach was compared with the k-means algorithm. Tables 2 and 3 display the results obtained for the cases of three and five cluster centers, respectively.

Note that, on an average, for the case of three-cluster centers, the k-means algorithm required 31.65% of the CPU time consumed by the proposed approach, but the quality of the solution (with respect to the CP1 objective values) was significantly inferior being worse (greater) by a factor of 4.862. Indeed, the heuristic at node zero itself uniformly dominated the k-means algorithm, determining an objective function value that is 11.56% better (lesser) on an average, while consuming only 12.05% of CPU time. Similarly, in the case of five-cluster centers, the k-means algorithm required 22.76% of the time taken by the proposed exact approach, but produced a solution that was greater by a factor of 4.34. Again, the solution obtained by our method at node zero itself dominated the k-means solution, improving it on an average by 15.66%, while consuming only 11.45% of the CPU time required. Furthermore, the UB_0/LB^* column in Tables 2 and 3 records the LP-IP gap having an average value of 1.90 and 1.66, for three and five cluster centers, respectively. Also, a comparison of this ratio with UB_0/LB_0 reflects the extent of improvement in the final objective value attained (LB^*) versus the node zero incumbent value LB_0.

Note that the performance of the branch-and-bound algorithm is influenced by three factors: First, choosing the best model formulation among variations of Problem CP1.4; second, selecting an appropriate strategy to ameliorate the effects of symmetry and third, implementing a judicious

Table 2. Relative performance of the proposed optimization approach versus the k-means algorithm, measured in terms of different parameters, for three cluster centers

Data sets	Parameters							
	$\frac{UB_0}{LB_0}$	$\frac{UB_0}{LB^*}$	$\frac{Z_0^*}{Z^*}$	$\frac{Z_{k-\text{means}}^*}{Z^*}$	CPU^* (s)	$\text{CPU}_{k-\text{means}}$(s)	$\frac{\text{CPU}^*}{\text{CPU}_{k\text{-means}}}$	$\frac{\text{CPU}_0}{\text{CPU}_{k\text{-means}}}$
1	4.67	1.44	3.15	3.41	0.20	0.14	1.428	0.052
2	6.15	2.02	5.43	7.688	0.40	0.20	2.0	0.074
3	6.72	2.30	4.55	4.73	1.28	0.30	4.267	0.195
4	5.09	1.87	4.08	3.62	0.90	0.24	3.75	0.161
Averages	5.65	1.90	4.30	4.862	0.695	0.22	2.861	0.1205

Table 3. Relative performance of the proposed optimization approach versus the *k*-means algorithm, measured in terms of different parameters, for five cluster centers

Data Sets	Parameters							
	$\frac{UB_0}{LB_0}$	$\frac{UB_0}{LB^*}$	$\frac{Z_0^*}{Z^*}$	$\frac{Z_{k-\text{means}}^*}{Z^*}$	CPU^* (s)	$\text{CPU}_{k-\text{means}}$(s)	$\frac{\text{CPU}^*}{\text{CPU}_{k-\text{means}}}$	$\frac{\text{CPU}_0}{\text{CPU}_{k-\text{means}}}$
1	3.73	1.12	1.55	1.80	0.355	0.14	2.535	0.089
2	5.85	2.09	6.15	4.57	0.59	0.22	2.68	0.10
3	7.71	2.35	4.50	6.22	2.10	0.28	7.5	0.16
4	4.20	1.08	2.44	4.78	1.70	0.44	3.86	0.109
Averages	5.37	1.66	3.66	4.34	1.186	0.27	4.14	0.1145

branching mechanism. Hence, prior to evaluating the robustness of the proposed approach on relatively larger problem instances, several computational tests were performed on some sample problems to ascertain the effects of the foregoing three features and thereby, compose a suitable algorithmic approach. The results of these various experimental runs are recorded in Tables 4–7.

To begin with, Table 4 displays the comparative results obtained for the two proposed symmetry-defeating strategies. For this purpose, four randomly generated sample problems involving three clusters, and having the number of data points and attributes (dimension) as indicated in Table 4 were solved, with the model variations being CP1.4 without (16f, g), but including either (13a, b), or (14), or neither. Also, in Table 4, the values in the parentheses recorded for each cell indicate the two-tuple $(UB_0/LB_0, \text{CPU}^*)$, i.e., the gap ratio at node zero and the total CPU time. Based on the results obtained, note that Problem CP1.4 with (14) included obtains an average UB_0/LB_0 gap ratio of 10.1 (which is marginally worse than that for CP1.4 with (13a, b) included) but consumes the least amount of CPU time. Evidently, attempting to identify distinct clusters based on

Table 4. Variations of Problem CP1.4 to test the effectiveness of the different symmetry defeating strategies, measured in terms of gap ratio at node zero and the CPU time

Problem type	Data Set				Averages
	1	2	3	4	
	(250, 4)	(250, 6)	(500, 4)	(500, 6)	
CP1.4 without (16f, g) and only (13) included	(9.05, 144.11)	(8.74, 288.42)	(6.87, 312.62)	(14.67, 421.6)	(9.83, 280.68)
CP1.4 without (16f, g) and only (14) included	(8.75, 128.40)	(8.74, 269.10)	(8.08, 279.05)	(14.88, 387.2)	(10.1, 271.05)
CP1.4 without (16f, g) and neither (13, 14)	(9.24, 144.45)	(11.67, 298.4)	(12.43, 344.5)	(15.45, 474.5)	(12.19, 315.4)

Table 5. Variations of Problem CP1.4 to test the effectiveness of the different bounding constraints measured in terms of the gap ratio at node zero and the CPU time

Problem type	Data Set				Averages
	1	2	3	4	
	(250, 4)	(250, 6)	(500, 4)	(500, 6)	
CP1.4 with only (16f) included	(9.24, 4.36, 138.76)	(10.3, 5.75, 266.0)	(8.55, 4.08, 332.62)	(15.45, 7.88, 442.6)	(10.9, 5.51, 295.0)
CP1.4 with both (16f, g) included	(7.44, 2.84, 116.81)	(8.74, 3.90, 269.10)	(7.90, 3.37, 258.2)	(12.33, 4.97, 369.3)	(9.1, 3.77, 253.35)
CP1.4 with neither (16f, g) included	(9.24,4.36, 144.45)	(11.67, 6.44, 298.4)	(12.43, 8.15, 344.5)	(15.45, 8.03, 474.5)	(12.19, 6.74, 315.4)

Table 6. Performance of the different branching strategies, measured in terms of the number of nodes enumerated and the CPU time

Branching rule	Data set				Averages
	1	2	3	4	
	(250, 4)	(250, 6)	(500, 4)	(500, 6)	
CP1.4 with branching strategy (17a, b)	(1068, 113.2)	(1454, 256.0)	(2455, 340.45)	(3580, 462.4)	(2139, 293.0)
CP1.4 with SOS branching strategy	(866, 116.81)	(974, 269.1)	(1375, 258.2)	(2421, 369.3)	(1409, 271.0)

Table 7. The performance of CP1.4 via the default strategies of CPLEX-MIP 8.1.0 and CP1.3 via BARON, measured in terms of CPU time

Problem type	Data set				Averages
	1	2	3	4	
	(250, 4)	(250, 6)	(500, 4)	(500, 6)	
CP1.4 via CPLEX-MIP 8.1.0 default settings	263.44	360.83	577.0	711.25	478.13
CP1 with (14) via the BARON global optimizer	388.45	344.67	649.07	697.7	468.61
$\frac{Z^*_{\text{BARON}}}{Z^*}$	1.0	2.43	1.79	3.45	2.16

the allocation of a set of most dispersed points serves to provide an effective symmetry-defeating strategy, and is the one we propose to implement henceforth. Note that ignoring the effects of symmetry takes 19.36% greater effort, and is clearly not advisable.

Next, the efficacy of including the constraints (16f) and (16g) was tested. Here, the performance of CP1.4 measured according to the gap ratio at node zero, both with respect to the node zero incumbent value LB_0 and the optimal solution value LB^*, as well as the CPU time consumed was examined for the three cases corresponding to using only (16f), using (16f) and (16g), and using neither. (Partial results for the most latter case are given in Table 4). Table 5 displays the results obtained for these various formulations as a three-tuple given by $(UB_0/LB_0, UB_0/LB^*, \text{CPU}^*)$, and provides a measure of the quality of the LP relaxation. Observe that including constraints (16f) and (16g) leads to a decrease in the node zero gap ratio, with respect to both the node zero incumbent value LB_0 and the optimal solution value LB^*, as well as reduces the overall CPU effort. In comparison with the case wherein neither of (16f, g) was included, these values decreased by 25.34, 44.06, and 19.67%, respectively. Likewise, there is a decrement of 16.51, 31.57, and 14.1% in these respective values, in comparison with the case when only (16f) is present. Hence, we recommend incorporating both (16f) and (16g) in the model formulation.

The third test performed was to evaluate the two alternative branching strategies proposed in Section 2, based on imposing the dichotomy $w_{uv} = 0$

or 1 on a single variable as identified by (17a, b), or based on the SOS partitioning scheme as designated in Remark 3. The performance of these branching strategies is reported in Table 6 in terms of the two-tuple: (number of nodes enumerated in the branch-and-bound tree, CPU time taken to determine an optimal solution). The results obtained appear to indicate that the partitioning scheme given by (17a, b) is more efficient for smaller sized problems, but the SOS branching strategy begins to dominate as the size of the problem increases.

Although the SOS branching strategy uniformly dominates in terms of the number of nodes enumerated, the effort required at each node is greater and hence, for the relatively smaller sized problems, more CPU time is taken to determine an optimal solution. However, as the problem size increases, the number of nodes enumerated is considerably larger for the branching strategy (17a, b) as compared with the SOS branching method to the extent that the SOS branching scheme begins to dominate. Naturally, the solution of larger sized problems more effectively is of greater concern, and so we recommend the use of the SOS branching strategy.

Finally, for the purpose of comparison, a computational study was performed to test the efficacy of solving the enhanced model formulation CP1.4 directly by the commercial software CPLEX-MIP 8.1.0 using its default settings. Furthermore, as a point of interest, the commercial global optimizer BARON (refer Sahinidis, 1996, 1999–2000) was utilized to directly solve the original model CP1 augmented by the symmetry-defeating constraints (14). Let us denote the best objective function value obtained by solving CP1 with (14) via BARON as Z^*_{BARON}. Table 7 displays the CPU times obtained for each of these cases, and the ratios of the final objective function values. Comparing the results displayed in Table 7 with those in Table 6, it can be seen that both CP1.4 solved directly by CPLEX-MIP 8.1.0 as well as CP1 with (14) solved via BARON consume a significantly greater CPU time for larger problem instances, as compared with using the proposed branch-and-bound algorithm. Indeed, in those instances where solving the nonlinear program CP1 with (14) via BARON dominated in terms of CPU time, it terminated at a significantly inferior local optimal solution (as recorded by the $Z_{\mathrm{BARON}}{}^*/Z^*$ values). Assimilating the information given in Tables 4–7, we conclude that Problem CP1.4 including the constraints (16f, g) along with the symmetry-defeating mechanism given by (14), and solved via the proposed branch-and-bound algorithm utilizing the SOS branching scheme affords the most viable composition of the tested strategies for solving relatively large instances of the hard clustering problem.

To reinforce this and to establish the robustness of the proposed approach, we solved several additional problems of larger sizes, and also

compared the results obtained with those produced by the popular k-means algorithm. The number of data points in these test instances was varied from 250 to 1000 in steps of 250, and the dimension of the space was varied from two to eight, in steps of two, thereby leading to a total of $4 \times 4 = 16$ test problems, with the smallest data set having 250 points in a two-dimensional space, and the largest problem having 1000 points in an eight-dimensional space. The number of clusters (c) for each case was taken to be either three (Table 8) or five (Table 9).

From the results displayed in Tables 8 and 9, note that the k-means algorithm requires a significantly lesser CPU time as compared with the proposed exact approach, but the best solution produced by the k-means algorithm is also substantially inferior. However, the node zero heuristic solution produced by the proposed approach uniformly dominates the k-means solution with respect to both quality and effort in most of the problem instances, with the exceptions being shaded in the rows of Tables 8 and 9. On an average, to obtain a feasible solution to Problem CP1 based on the node zero analysis alone, the CPU time required is on an average 17.2% lesser than for the k-means algorithm, yet the quality of the solution is 13.3% better in terms of the objective function value for the three cluster center case. A similar result holds true for the case of five cluster centers. Using a more sophisticated heuristic than the one advocated in Remark 4, or improving this solution by appending some steps of a suitable meta-heuristic approach such as the genetic algorithm or simulated annealing,

Table 8. Results for the proposed approach and the k-means algorithm for large problem instances having three cluster centers

Data sets	Parameters							
	$\frac{UB_0}{LB_0}$	$\frac{UB_0}{LB^*}$	$\frac{Z_0^*}{Z^*}$	$\frac{Z^*_{k\text{-means}}}{Z^*}$	CPU* (s)	CPU$_{k\text{-means}}$(s)	$\frac{\text{CPU}_{\text{CP1.4}}}{\text{CPU}_{k\text{-means}}}$	$\frac{\text{CPU}_0}{\text{CPU}_{k\text{-means}}}$
(250, 2)	6.72	3.17	2.45	3.34	35.411	3.314	10.685	0.635
(500, 2)	7.19	1.85	2.61	3.58	74.554	5.335	13.974	0.504
(750, 2)	25.4	4.44	12.9	9.00	150.30	11.027	13.630	0.978
(1000, 2)	13.1	2.29	4.70	6.62	180.87	17.578	10.289	0.844
(250, 4)	15.8	2.76	5.66	8.02	139.19	4.58	30.392	0.751
(500, 4)	26.5	4.64	11.8	11.07	246.60	8.572	28.768	1.779
(750, 4)	8.89	1.55	3.21	4.45	377.67	10.208	36.997	0.615
(1000, 4)	32.2	5.64	11.4	16.4	708.10	17.578	40.283	0.497
(250, 6)	16.5	3.89	5.88	8.34	384.33	23.801	16.147	0.555
(500, 6)	29.9	5.23	13.2	12.54	466.48	18.547	25.151	1.961
(750, 6)	21.5	3.76	7.64	10.9	485.43	14.602	33.244	0.783
(1000, 6)	7.09	2.24	2.58	3.53	508.38	18.790	27.056	0.634
(250, 8)	31.0	5.43	15.5	11.2	392.33	17.578	22.316	0.932
(500, 8)	4.21	2.73	1.57	2.06	945.16	20.316	46.523	0.589
(750, 8)	32.6	5.71	11.5	10.6	1181.1	32.106	36.787	1.089
(1000, 8)	23.7	4.15	8.40	12.0	1425.3	39.925	35.701	0.603
Averages	18.9	3.71	7.56	8.72	481.33	16.491	26.746	0.859

Table 9. Results for the proposed approach and the k-means algorithms for large problem instances having five cluser centers

Data sets	Parameters							
	$\frac{UB_0}{LB_0}$	$\frac{UB_0}{LB^*}$	$\frac{Z_0^*}{Z^*}$	$\frac{Z^*_{k\text{-means}}}{Z^*}$	CPU* (s)	CPU$_{k\text{-means}}$ (s)	$\frac{\text{CPU}_{\text{CP1.4}}}{\text{CPU}_{k\text{-means}}}$	$\frac{\text{CPU}_0}{\text{CPU}_{k\text{-means}}}$
(250, 2)	12.4	2.17	2.29	2.77	28.383	1.793	15.829	0.681
(500, 2)	12.1	3.11	5.33	6.43	73.495	2.887	25.457	0.729
(750, 2)	8.26	1.44	8.51	7.85	127.89	5.967	21.433	1.106
(1000, 2)	1.37	1.23	8.07	9.74	191.35	16.092	11.891	0.574
(250, 4)	2.98	1.52	6.80	8.21	116.81	4.638	25.185	0.722
(500, 4)	3.13	1.54	5.13	4.99	258.20	5.524	46.741	0.955
(750, 4)	14.2	3.48	8.21	9.91	349.13	12.092	28.873	0.661
(1000, 4)	1.21	1.21	11.2	13.6	360.20	12.879	27.968	0.935
(250, 6)	1.71	1.29	3.89	4.70	269.10	10.036	26.813	0.997
(500, 6)	25.2	4.41	8.13	7.40	369.33	7.901	46.744	1.716
(750, 6)	6.83	1.19	7.70	9.29	475.24	10.167	46.743	0.736
(1000, 6)	1.23	1.21	5.21	6.29	491.35	26.092	18.831	0.414
(250, 8)	2.47	1.43	6.79	8.19	313.84	10.993	28.549	0.915
(500, 8)	25.0	4.37	4.71	5.69	812.03	17.373	46.740	0.642
(750, 8)	1.05	1.05	10.8	13.1	1009.8	21.604	46.741	0.614
(1000, 8)	11.6	3.03	8.83	9.45	1257.5	26.905	46.741	0.785
Averages	13.1	2.10	6.97	7.97	406.48	12.058	33.708	0.611

might lead to a more effective procedure. Moreover, utilizing a better approximation to the convex hull of data points in the model formulation could lead to a further improvement in the performance of both the exact and heuristic routines. We recommend these investigations for future research.

5. Summary and Extensions for Further Research

This research effort addresses the issue of determining a global optimum to the hard clustering problem, where the objective function seeks to minimize the total squared (Euclidean) distance from each data point to the center of the cluster to which it is assigned. A series of enhanced reformulations of this problem were presented, augmented by valid inequalities and RLT-based constraints. A specialized branch-and-bound algorithm was designed for the resulting equivalent 0-1 mixed-integer programming problem. Several computational experiments were performed using standard data sets as well as synthetically generated test cases, to explore the efficacy of including the different proposed model enhancement strategies, as well as to study the effectiveness of the heuristic scheme implemented at the root node. Furthermore, this performance was compared with the k-means algorithm (see Forgy, 1966; McQueen, 1967) that is popularly used in the literature on this

topic. The results support the robustness of the proposed approach, and exhibit its superiority over the k-means algorithm (even as a heuristic based on the node zero analysis). In particular, the RLT-enhanced model CP1.4 coupled with a valid symmetry-defeating strategy, and solved via the proposed branch-and-bound algorithm using an SOS branching mechanism yielded the best combination of the strategies tested, and is recommended for solving the hard clustering problem. Note that in practice, cluster analysis problems often involve very large data sets, and therefore, good heuristic procedures are essential for handling such problem instances. Our research suggests that designing heuristic methods based on constructs that are borrowed from strong effective exact procedures might be a prudent approach.

Finally, note that the number of cluster centers is introduced as a fixed, external parameter into the optimization model, as opposed to finding an optimal number of clusters, given a certain data set. A decision criterion to determine an optimal number of clusters in hierarchical clustering was advocated by Jung et al. (2003). Our work could be extended to accommodate this feature as well. As another possible extension, an alternative idea that one could use to address the issue of symmetry (see Remark 2), as well as to develop an effective heuristic procedure is as follows.

Define a *cluster vector* v_r, for any index r, to have n components, with the ith component being 1 if the point a_i is assigned to the particular cluster, and 0 otherwise.

Let

$$V_r = \{i : (v_r)_i = 1\}. \tag{18}$$

Given a cluster vector v_r with the associated set of assigned points V_r, the optimal center location z has components given via (2b) as

$$z_k^* = \sum_{i \in V_r} a_{ik} \Big/ |V_r|. \tag{19}$$

The objective cost term associated with this cluster vector, C_r, is given as follows, using (3) and (19).

$$C_r \equiv \sum_{i \in V_r} \sum_{k=1}^{s} (z_k^* - a_{ik})^2 = \sum_{i \in V_r} \sum_{k=1}^{s} a_{ik}^2 - \frac{1}{|V_r|} \sum_k \left\{ \sum_{i \in V_r} a_{ik} \right\}^2. \tag{20}$$

Suppose that we have generated several potential cluster vectors indexed by $r = 1, 2, \ldots, R$, based on various covers of the scatter of data points. Then we can solve the following set partitioning problem (SPP), where e is a vector of n ones.

$$\textbf{SPP :} \text{ Minimize} \sum_{r=1}^{R} C_r x_r \tag{21a}$$

$$\text{subject to} \sum_{r=1}^{R} v_r x_r = e \tag{21b}$$

$$x \text{ binary.} \tag{21c}$$

Note that feasibility of (21a)–(21) can be assured by including within it a known partition of the data points into suitable cluster vectors. Furthermore, having solved this, we could try generating additional clusters that might yield a negative reduced cost for the LP relaxation $\overline{\mathbf{SPP}}$ (and hence, perhaps lead to an improved IP solution) by monitoring the following reduced cost expression, as points are added to V_r, where $\bar{\pi}$ is an optimal dual solution vector associated with (21b).

$$C_r - \bar{\pi} v_r = \sum_{k=1}^{s} \left\{ \sum_{i \in V_r} a_{ik}^2 - \frac{1}{|V_r|} \left[\sum_{i \in V_r} a_{ik} \right]^2 \right\} - \sum_{i \in V_r} \bar{\pi}_i. \tag{22}$$

Conceivably, some genetic algorithmic concepts could be applied to generate such advantageous members from the population of cluster vectors. Turning this into an effective heuristic scheme is a topic that is recommended for future research. Note that this algorithmic process could also possibly be converted into an exact algorithm by using branch-and-price concepts, although one would need to contend in this context with a nonconvex objective function, which would be problematic.

Acknowledgement

This work has been supported by the *National Science Foundation* under Grant No. DMI - 0094462. The authors also thank Dr. Nick Sahinidis of the $\sum$-Optimization Group at the University of Illinois, Urbana Champaign, for permitting the use of the BARON software.

References

1. Al-Sultan, K.S. and Khan, M.M. (1996), Computational experience on four algorithms for the hard clustering problem, *Pattern Recognition Letters* 17, 295–308.
2. Balas, E. (1988), On the convex hull of the union of certain polyhedra, *Operations Research Letters* 7(6), 279–283.
3. Bhuyan, J.N., Raghavan, V.V. and Elayavalli, V.K. (1991), Genetic algorithm for clustering with an ordered representation, *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Diego, CA.
4. Chazelle, B. (1991), An optimal convex hull algorithm and new results on cuttings, *Annual Symposium on Foundations of Computer Science*, 29–38.
5. Dubes, R.C. (1987), How many clusters are best? – an experiment, *Pattern Recognition* 20, 645–663.

6. Forgy, E.W. (1966), Cluster analysis of multivariate data: efficiency versus interpretability of classification, *Biometric Society Meetings*, Riverside, CA, *Abstract in Biometrics* 21, 768.
7. Groetschel, M. and Wakabayashi, Y. (1989), Cutting plane algorithm for a clustering problem, *Mathematical Programming Series B* 45(1), 59–96.
8. Hartigan, J.A. (1975), *Clustering Algorithms*, John Wiley and Sons, New York, N.Y.
9. Jensen, R.E. (1969), A dynamic programming algorithm for cluster analysis, *Operations Research* 17, 1034–1057.
10. Jung, Y., Park, H., Du, Z. and Drake, B.L. (2003), A decision criterion for the optimal number of clusters in hierarchical clustering, *Journal of Global Optimization* 25, 91–111.
11. Klaapper, A. (1987), Lower bound on the complexity of the convex hull problem for simply polyhedra, *Information Processing Letters* 25(3), 159–161.
12. Klein, R.W. and Dubes, R.C. (1989), Experiments in projection and clustering by simulated annealing, *Pattern Recognition* 22, 213–220.
13. Koontz, W.L., Narendra, P.M. and Fukunaga, K. (1975), A branch-and-bound clustering algorithm, *IEEE Transactions on Computing* 23, 908–914.
14. Krovi, R. (1992), Genetic algorithm for clustering: a preliminary investigation, *Proceedings of the 25th Hawaii International Conference on Systems Sciences*, pp. 540–544.
15. Leung, Y., Zhang, J. and Xu, Z. (1997), Neural networks for convex hull computation, *IEEE Transactions on Neural Networks* 8(3), 601–611.
16. Lukashin, A.V. and Fuchs, R. (2000), Analysis of temporal gene expression profiles: clustering by simulated annealing and determining the optimal number of clusters, *Bioinformatics* 17(5), 405–414.
17. Manber, U. (1989), *Introduction to Algorithms: A Creative Approach*, Addison-Wesley Publishing Company, Reading, MA.
18. Mangiameli, P., Chen, K.S. and West, D. (1996), A comparison of SOM neural network and hierarchical clustering methods, *European Journal of Operations Research* 93, 402–417.
19. McQueen, J.B. (1967), Some methods of classification and analysis of multivariate observations, *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press, Berkeley, CA, pp. 281–197.
20. Mulvey, J.M. and Crowder, H.P. (1979), Cluster analysis: An application of lagrangian relaxation, *Management Science* 25(4), 329–340.
21. Rao, M.R. (1971), Cluster analysis and mathematical programming, *Journal of American Statistical Association* 66, 622–626.
22. Sahinidis, N.V. (1996), BARON: A general purpose global optimization software package, *Journal of Global Optimization* 8, 201–205.
23. Sahinidis, N.V. (1999–2000), *BARON: Branch-and-Reduce Optimization Navigator*, *User's Manual*, *Version 4.0*, Available for download at http://archimides.scs.uiuc.edu/baron.html.
24. Selim, S.Z. (1982), A global algorithm for the clustering problem, *Presentation at the ORSA/TIMS Joint Meeting*, San Diego, CA.
25. Selim, S.Z. and Al-Sultan, K.S. (1991), A simulated annealing algorithm for the hard clustering problem, *Pattern Recognition* 24, 1003–1008.
26. Sherali, H.D. and Adams, W.P. (1990), A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems, *SIAM Journal on Discrete Mathematics* 3(3), 411–430.
27. Sherali, H.D. and Adams, W.P. (1994), A hierarchy of relaxations and convex hull characterizations for mixed-integer zero-one programming problems, *Discrete Applied Mathematics* 52, 83–106.

28. Sherali, H.D. and Adams, W.P. (1999), Reformulation-linearization techniques for discrete optimization problems, In: Du, D.Z. and Pardalos, P.M. (eds.), *Handbook of Combinatorial Optimization* 1, Kluwer Academic Publishers, pp. 479–532.
29. Sherali, H.D. and Smith, J.C. (2001), Improving discrete model representations via symmetry considerations, *Management Science* 47(10), 1396–1407.
30. Sherali, H.D. and Tuncbilek, C. (1992), A global optimization algorithm for polynomial programming problems using a reformulation-linearization technique, *Journal of Global Optimization* 2, 101–112.
31. Sherali, H.D. and Tuncbilek, C. (1995), A reformulation-convexification approach for solving nonconvex quadratic programming problems, *Journal of Global Optimization* 7, 1–31.
32. Späth, H. (1980), *Cluster Analysis Algorithms for Data Reduction and Classification of Objects*, John Wiley and Sons, New York, NY.
33. Sultan, M., Wigle, D.A., Cumbaa, C.A., Maziarz, M., Glasgow, J., Tsao, M.S. and Jurisica, I. (2002), Binary tree-structured vector quantization approach to clustering and visualizing microarray data, *Bioinformatics* 18(1), 111–119.
34. Vinod, H.D. (1969), Integer programming and the theory of grouping, *Journal of American Statistical Society* 64, 506–519.
35. Ward, J.H. Jr. (1963), Hierarchical grouping to optimize an objective function, *Journal of American Statistical Society* 58, 236–244.