Multiobjective Clustering with Metaheuristic Optimization Technology

Rafael Caballero

Department of Applied Economics (Mathematics), University of Málaga, Campus El Ejido s./n., 29071 Málaga, Spain, rafael.caballero@uma.es

Manuel Laguna

Leeds School of Business, 419 UCB, University of Colorado, Boulder, Colorado 80309, USA, laguna@colorado.edu

Rafael Martí

Department of Statistics and Operational Research, University of Valencia, Dr. Moliner 50, 46100 Burjassot (Valencia), Spain, rafael.marti@uv.es

Julián Molina

Department of Applied Economics (Mathematics), University of Málaga, Campus El Ejido s./n., 29071 Málaga, Spain, julian.molina@uma.es

Abstract – We develop a metaheuristic procedure for multiobjective clustering problems. Our goal is to find good approximations of the efficient frontier for this class of problems and provide a means for improving decision making in multiple areas of application and in particular those related to marketing. The procedure is based on the tabu and scatter search methodologies. Clustering problems have been the subject of numerous studies; however, most of the work has focused on single-objective problems. Clustering using multiple criteria and/or multiple data sources has received limited attention in the OR and marketing literature. Our procedure is general and tackles several problems classes within this area of combinatorial data analysis. We conduct extensive experimentation with both artificial and real data (in a marketing-segmentation problem) to show the effectiveness of the proposed procedure.

Keywords: multiobjective optimization, scatter and tabu search, combinatorial data analysis, partitioning, market segementation.

1 Introduction

We study optimization problems associated with cluster analysis, within the general area of combinatorial data analysis. In particular, we are interested in partitioning problems for cluster analysis that requires the simultaneous optimization of more than one objective function. Clustering problems have a wide range of application areas and are particularly important in marketing studies. The following example illustrates the need for modeling clustering problems in a multiobjective optimization framework.

Consider a situation in which a firm would like to identify consumer segments that have different levels of proneness to deals. The firm conjecture is that there may be customers prone to deals in general and some other customers prone to specific type of deals. Suppose that the firm collects survey data (e.g., Likert-scale measurements) regarding the proneness of customers to deals, such as coupons, buy-one-get-one-free, free gifts and rebates. The firm has data on demographics, such as age, household income, education and gender. Also, additional data are available on customers' behaviors pertaining deal responsiveness, such as coupon redemption frequency and quantity of sale items purchased. With these data, the firm could apply standard (single-criterion) clustering techniques to find clusters that are homogenous with respect to the customers' demographics. Homogeneity is given by the selection of the objective function to be optimized when solving the associated clustering problem. As we discuss later, there are a number of criteria that could be used to obtain homogenous clusters. The firm, instead, may choose to cluster customers in order to maximize the explained variation in the actual customers' responses to deals. It is possible, however, that neither of these clustering schemes turn out to be useful to the firm that might be looking for clusters that are both homogenous according to some criterion and at the same time explain the variation related to one or more dependent variables. We come back to this situation in our computational experiments, where we use data collected by Lichtenstein, Burton and Netemeyer (1997), who studied customers' proneness to eight different types of sales promotions.

We assume the existence of an $n \times n$ symmetric matrix $\mathbf{A} = \{a_{ij}\}$ such that $a_{ij} \geq 0$ for $i \neq j$ and where the values of the elements in the main diagonal are irrelevant. The values in $\mathbf{A}$ reflect the level of dissimilarity between each pair of objects in a data set, such that the larger a_{ij} is the more dissimilar object i is from object j (and vice versa). Cluster analysis methods may be hierarchical or nonhierarchical. Our work focuses on nonhierarchical (or partitioning) methods.

Partitioning methods consist of separating objects into K distinct clusters C_k of size n_k. A feasible partition of clusters satisfies the following conditions:

$C_k \neq \varnothing$ for $1 \leq k \leq K$

$C_h \cap C_k = \varnothing$ for $1 \leq h < k \leq K$

$C_1 \cup C_2 \cup \ldots \cup C_K$ = Collection of all objects

We consider two main classes of multiobjective partitioning problems: 1) partitioning of objects using one partitioning criterion but multiple dissimilarity matrices and 2) partitioning of objects using one dissimilarity matrix but more than one partitioning criteria. (Our computational experiments also include partitioning problems using both multiple criteria and multiple dissimilarity matrices.) As mentioned by Brusco and Cradit (2005), "The spectrum of possible indices for obtaining a partition based on proximity data is limited only by the analyst's imagination; however, most indices correspond to either the sum of dissimilarity elements within clusters, or the maximum dissimilarity element (diameter) within clusters." Therefore, following Brusco and Stahl (2005), we will focus on partitioning criteria that consist of minimizing a combination of the following four functions:

Partition diameter: $$f_1 = \max_{k=1,\ldots,K} \left(\max_{(i<j) \in C_k} \left(a_{ij} \right) \right)$$

Unadjusted within-cluster dissimilarity: $$f_2 = \sum_{k=1}^{K} \sum_{(i<j) \in C_k} a_{ij}$$

Adjusted within-cluster dissimilarity: $$f_3 = \sum_{k=1}^{K} \left(\frac{\sum_{(i<j)\in C_k} a_{ij}}{n_k} \right)$$

Average within-cluster dissimilarity: $$f_4 = \sum_{k=1}^{K} \left(\frac{\sum_{(i<j)\in C_k} a_{ij}}{n_k (n_k - 1)/2} \right)$$

Minimizing the partition diameter entails the calculation of the diameter for each cluster. The cluster diameter is the maximum dissimilarity between two objects that belong to the same cluster. This criterion is quite different from the other three, which are based on adding the dissimilarity values within each cluster. This is why a typical bi-criterion problem consists of finding a partition of objects that minimizes the partition diameter and one of the other functions based on an aggregate within-cluster dissimilarity measure.

1.1 Partitioning of Objects Using Multiple Criteria

Minimizing the diameter of a partition is conceptually different from minimizing a function based on aggregating the within-cluster dissimilarity values. Hence, it is not reasonable to expect that a partition that is optimal with respect to f_1 will be also optimal with respect to f_2, f_3 or f_4. In fact, even though criteria f_2, f_3 and f_4 are based on aggregating dissimilarity values, there is no guarantee of finding a single partition that optimizes all of them simultaneously. Consider, for example, using the dissimilarity matrix in Table 1 to solve a partitioning problem for $K = 2$.

	1	2	3	4
1	0	4	5	3
2	4	0	6	5
3	5	6	0	7
4	3	5	7	0

Table 1 Dissimilarity matrix

Table 2 shows the optimal solutions associated with each objective function under consideration.

Partition	f_1	f_2	f_3	f_4
{1, 3} {2, 4}	5*	10	5	10
{1, 4} {2, 3}	6	9*	4.5	9
{3} {1, 2, 4}	12	12	4*	4*

* Optimal objective function value

Table 2 Optimal partitions for Table 1 data

The first partition in Table 2 is optimal with respect to f_1, but is suboptimal with respect to all other criteria. Likewise, the second partition is optimal with respect to f_2 but suboptimal with respect to the other functions. The third partition is somewhat special in that it is optimal for two out of the four objective functions.

The challenge of this multiobjective optimization problem is finding efficient partitions for any combination of the criteria described above. As mentioned above, a typical problem in this context consists of formulating bipartite models to minimize f_1 and either f_2, f_3 or f_4 (Brusco and Cradit, 2005). These problems are not the only ones that have been addressed in the literature. For instance, Hansen and Delattre (1978) introduced f_1 to the cluster analysis literature and also suggested using the concept of *split* to find partitions of objects (Delattre and Hansen, 1980). The split of a cluster C_k is defined as the

minimum dissimilarity between an object in C_k and any other object not in C_k. The split may be used to either maximize its minimum value (Delattre and Hansen, 1980) or maximize the sum (Hansen, Jaumard and Frank, 1989).

1.2 Partitioning of Objects Using Multiple Dissimilarity Matrices

This problem arises when clustering of objects is done by considering more than one data set and it is not possible to aggregate these multiple sets into one. The market segmentation problem described by Krieger and Green (1996) is a typical case of partitioning using multiple dissimilarity matrices. The problem consists of finding homogenous clusters in order to plan targeted marketing efforts. If the clustering is based on a single data source, then traditional cluster analysis techniques can be applied. However, there are situations where the market segmentation is more effective if several data sources are used. For instance, Brusco and Stahl (2005) describe the case of a telecommunications company that would like to find clusters of business customers based on their demographics and also based on current satisfaction and their willingness to switch providers. While the two sources of data could be aggregated to perform a traditional cluster analysis, the telecommunications company is more interested in identifying homogenous clusters based on demographics while explaining the variation in current satisfaction levels and willingness to switch. The appropriate approach is therefore to formulate the problem as a multiobjective partitioning of objectives using multiple dissimilarity matrices (i.e., firm demographics and customer satisfaction/willingness to switch). Note that this situation is similar to the one we describe in the introduction of this paper to motivate the need for multiobjective clustering in marketing.

In general terms, the problem is one in which performance drivers are known (first source of data) along with their performance measures (second source of data) and it is desired to identify K clusters that are homogenous with respect to the performance drivers and that at the same time they help to explain the variation on the performance measures. A typical objective function in this situation is to minimize the total adjusted within-cluster dissimilarity for the L sources of data:

$$\text{minimize} \qquad f_3^l = \sum_{k=1}^{K} \left(\frac{\sum_{(i<j)\in C_k} a_{ij}^l}{n_k} \right) \qquad \text{for } l = 1, \ldots, L$$

where a_{ij}^l = dissimilarity of objects *i* and *j* according to data source *l*. Dissimilarity typically consists of the Euclidean distance between two firms as calculated with the tuple of corresponding measures. In this context, it is also usual to maximize the proportion of explained variation, which is calculated by reference to the total sum of squares (*TSS*):

$$TSS_l = \frac{\sum_{i<j} a_{ij}^l}{n} \qquad \text{for } l = 1, \ldots, L$$

The multiobjective optimization problem becomes:

$$\text{maximize} \qquad g_3^l = \frac{f_3^l}{TSS_l} \qquad \text{for } l = 1, \ldots, L$$

Note that g_3^l is bounded between zero and one. The extreme values occur at the trivial solutions that consist of putting each object in a different cluster ($K = n$ and $g_3^l = 0$) and putting all objects in the same cluster ($K = 1$ and $g_3^l = 1$). We tackle this particular problem in the experiments presented in section 4.3.

2 Pareto Approach

Multiobjective partitioning problems have been approached in two general ways: 1) using preemptive (or priority) weights to assign priorities to the objective functions, or 2) using weights to aggregate the objective functions into a single one. Suppose that it is desired to find partitions that minimize both f_1 and f_2. Under the preemptive-weight approach, the optimization problem may be formulated as follows:

$$\text{Min} \quad P_1 f_1 + P_2 f_2$$

In this formulation, P_1 and P_2 are preemptive weights that determine the priority of the objective functions. Objective functions of higher priority are optimized first follow by those with lower priority. In other words, the preemptive weights create a hierarchy of objective functions. Under this approach, lower priority objective functions are not allowed to alter the level of attainment of the objective functions at a higher priority. In our example, we would first find f_1^*, the optimal value for the objective function with the higher priority. Then, a search for a partition that minimizes f_2 is conducted in such a way that $f_1 \leq f_1^*$. In order to generate more than one solution, the diameter restriction may be relaxed (from its optimal value), resulting in the multiobjective approach known as the ε-constraint method. The main disadvantage of this approach is that in order to find an approximation of the Pareto frontier, a potentially large number of problems of the following form must be solved:

$$\text{Min} \quad f_v$$

s.t.

$$f_i \leq \varepsilon \qquad i = 1, \ldots, p,\ i \neq v$$

The task becomes computationally impractical as the number of objective functions (p) increases. This is the approach followed by Brusco and Cradit (2005) and that we use for comparison purposes in section 4.1.

In the second approach, the multiobjective optimization problem is transformed to a single-objective function problem of the form:

$$\text{Min} \quad \sum_{i=1}^{p} w_i f_i$$

In this model, w_i represents the weight assigned to the objective function f_i. Approximating the Pareto frontier with this approach entails the solution of this problem with different sets of weight values. Brusco and Stahl (2005) state the following in reference to this approach (page 85):

> "The principal limitation of multiobjective programming is the selection of an appropriate weighting scheme. The severity of this problem increases markedly when three or more criteria are considered."

This is the approach used by Brusco, Cradit and Stahl (2002) in their simulated annealing procedure. However, as we discuss in section 4.3, the approach has some severe limitations when dealing with combinatorial multiobjective optimization problems.

Our focus is to tackle the partitioning problems described above with the goal of finding an approximation $\hat{E}$ of the efficient frontier (or Pareto front) E. We do this by simultaneously considering all the objective functions in a given problem. We do not aggregate the objective functions nor do we prioritize them. The approach is based on searching in the direction of both the individual objective functions and *ideal points* constructed employing concepts from compromise programming. The search itself is conducted within the framework of metaheuristic optimization, as described in the next section.

3 Metaheuristic Search

Our metaheuristic search procedure is based on two well-known methodologies: tabu search (Glover and Laguna, 1997) and scatter search (Laguna and Martí, 2003). Our first application of this technology in the area of multiobjective optimization was related to nonlinear objective functions. In particular, we developed a procedure called SSPMO for nonlinear multiobjective optimization in continuous variables (Molina, et al. 2006). Our current development extends SSPMO to tackle the multiobjective clustering problems described above. We first provide a summarized description of SSPMO and then we discuss the search mechanisms that were specifically designed for the current context.

3.1 Summarized Description of SSPMO

SSPMO consists of two phases:

1. Generation of an initial set of efficient points through various tabu searches
2. Combination of solutions and updating of $\hat{E}$ via a scatter search

The first phase starts with $p + 1$ linked tabu searches, where p is the number of objective functions in the problem. The objective functions are considered one at a time (in arbitrary order). The tabu searches focus on finding the optimal solution to the single-objective optimization problems. Let x^1 be the last point visited at the end of the first tabu search. Then, another tabu search is launched, starting from x^1, to search for the optimal solution to the second single-objective optimization problem. This process is repeated until all the single-objective problems have been considered. A final tabu search is performed considering the first objective function one more time, starting from x^p. At this point, $\hat{E}$ contains the p solutions that best approximate the optimal solutions to the single-objective problems and all other efficient solutions found during the tabu searches.

Additional tabu searches are performed guided by a global criterion, as typically done in compromise programming (Zeleny, 1982). In particular, we use the L_∞ metric:

$$L_\infty(x) = \max_{i=1,\ldots p}\left\{ w_i \left(\frac{f_i^{ideal} - f_i(x)}{f_i^{ideal} - f_i^{anti-ideal}} \right) \right\}$$

where f_i^{ideal} is the best known value for objective function *i*, $f_i^{anti-ideal}$ is the worst possible outcome associated with objective *i*, and $f_i(x)$ is the value of the i^{th} objective function value associated with solution *x*. For a given set of weights ($w_i > 0$ and $\sum_i w_i = 1$) solution *x* is efficient if it minimizes $L_\infty(x)$. This property is the motivation behind the compromise-programming approach for the tabu searches that we perform beyond the first *p*+1. A new set of randomly generated weights is used for every tabu search. The first phase stops when *InitPhase* tabu searches are performed without changes in $\hat{E}$. It should be noted that the tabu search procedure used for this first phase is also applied as the improvement method in the second phase.

The scatter search phase is initiated immediately after the termination of the tabu search phase. The first step consists of building an initial reference set (*RefSet*) by drawing, without replacement, solutions from $\hat{E}$. Let *b* (*b* > *p*) be the size of *RefSet*. We first select the *p* solutions from $\hat{E}$ that best approximate the optimal values for each of the objective functions in the problem. Then, we sequentially choose *b*-*p* additional solutions from $\hat{E}$, where each selection is made with the criterion of maximizing the minimum distance between the chosen solution and the solutions currently in the reference set. In this case, distance is measured in the objective function space using a normalized L_∞ metric. Note that all the solutions in the initial *RefSet* are efficient (with respect to those in $\hat{E}$). We maintain this property throughout the search because we have determined that there is no benefit in combining solutions that have been shown to be dominated. An iteration of the scatter search phase starts with the application of

the combination method. The method consists of combining all pairs of reference solutions (i.e., all pairs of solutions in *RefSet*) in order to generate new solutions. Each combination of reference solutions yields 4 trials solutions. A tabu search is initiated from each of these trial solutions in an attempt to improve them. Trial solutions generated by the combination method and those visited during the tabu searches are tested for efficiency (i.e., they are tested for possible inclusion in $\hat{E}$). After the application of the combination and improvement methods, the *RefSet* is rebuilt following a process that is similar to the one described above for the initial reference set. In order to encourage a diversified exploration of the objective function space, we keep a record (*TabuRefSet*) of past reference solutions. No member of *TabuRefSet* can be chosen as a reference solution. Therefore, the rebuilding of *RefSet* is done with those solutions currently in $\hat{E}$ that do not belong to *TabuRefSet*.

3.2 Specific SSPMO Mechanisms for Clustering

A solution x is represented as a set of K objects, where each object is considered the "centroid" of a cluster. That is $x = \{x_1, \ldots, x_K\}$, where x_k ($1 \le x_k \le n$) is the centroid of cluster k. Objects that do not belong to x are assigned to clusters represented by the closest centroid. The assignment is done as follows:

$$C_k = \left\{ i : a'_{i,x_k} = \min_{h=1,\ldots,K} \left(a'_{i,x_h} \right) \right\} \qquad \text{for } k = 1, \ldots, K$$

For problems with one matrix $a'_{ij} = a_{ij}$. For problems with more than one dissimilarity matrix $a'_{ij} = \sum_l \lambda_l a^l_{ij}$, where λ_l is a weight assigned to data source l. In this case, we have one objective for each data source, i.e., $p = L$. Therefore, if we are performing the first p+1 search in the initial phase of the method, all lambda values are zero except for the one corresponding to the dissimilarity matrix under consideration, for which the lambda value is set to one. Afterwards, the lambda values are the same as

the weights used to create compromise solutions (i.e., $\lambda = w$). The justification for the choice of solution representation is presented in section 4.

For the tabu searches, we construct the neighborhood of a solution x by generating *NSize* distinct (r, q) pairs, where $q \notin x$, $1 \le q \le n$ and $1 \le r \le K$. A neighbor of x is such that x_r is replaced by q. The search moves to the best neighbor, where best is determined either by a single objective (during the p+1 initial searches) or by the L_∞ metric. If the current solution x is not dominated by any of its *NSize* neighbors, the solution is considered for membership in $\hat{E}$. Also, after the search moves to the best neighbor, x is added to the tabu list. Solutions remain in the tabu list for *TabuTenure* iterations. This is the simplest form of short-term memory tabu search that does not invoke aspiration level criteria.

The combination of two reference solutions is done as follows. Let x^1 and x^2 be the two reference solutions being combined. Then, the resulting trial solution x^t is such that

$$x_k^t = \begin{cases} x_k^1 & r < 0.5 \\ x_k^2 & r \ge 0.5 \end{cases}$$

where r is a random number between 0 and 1. If $x_k^t = x_h^t$ for $k \ne h$, the trial solution x^t is repaired by randomly selecting $q \notin x^t$ $(1 \le q \le n)$ and making $x_k^t = q$.

4 Computational Experiments

For our computational comparison we generated problem instances with the procedure suggested by Brusco and Cradit (2005) and Hansen and Delattre (1978). These authors created dissimilarity matrices of sizes n = 30, 60 and 90 by first generating points within a unit circle, then computing the Euclidean distance between pairs of points, and finally multiplying the distances by 100 and eliminating the fractional part. We have extended this set by adding two larger matrices with n = 120 and 150, respectively. The five matrices result in 35 instances when considering 7 different K values, ranging from

2 to 8. An additional set of 5 matrices with n = 30, 60, 90, 120 and 150 was created to provide data for experiments with multiple dissimilarity matrices. In this case, we use the pair of matrices of each size to provide two sources of data for each problem. As before, 7 K-values are considered, yielding 35 problem instances. All our experiments are performed on a Pentium machine with a single 2.4 GHz processor and 512 MB of RAM. Our C codes were compiled on Visual C++ 2005 Express Edition.

Brusco and Stahl (2005) provide exact procedures, based on the branch-and-bound methodology, for solving clustering problems with the single objectives f_1, f_2 and f_3. Their FORTRAN codes can be downloaded from http://garnet.acns.fsu.edu/~mbrusco/. No computer code is provided for the f_4 criterion. We use the dissimilarity matrix of size 30 to apply Brusco and Stahl's code to the 7 problem instances that correspond to K = 2 to 8. This was done to assess the capability of a state-of-the-art exact method for the class of problems that we are tackling. We limited the execution of the branch-and-bound code to 2 hours. The results are shown in Table 3. The first column shows the K value. The next three columns show the optimal solutions associated with each instance and each objective function. The last three columns show the time in CPU seconds.

The optimal solutions reported in Table 3 are the "extremes" of the efficient frontier for the multiobjective problem. These optimal values may be used to assess the quality of heuristic solutions when considering one objective function at a time. However, the branch and bound codes by Brusco and Stahl do not provide a way of finding efficient solution within these extreme points for problems with one data source and more than one objective function.

The results in Table 3 show that the branch and bound is an effective method for solving clustering problems with the f_1 criterion. Even for the largest problem instances with n = 120, the procedure finds and confirms the optimal solution in less than 15 CPU seconds. However, the procedure encounters difficulties with problem instances based on f_2 and f_3. The pattern observed in Table 3 extends to larger problem instances. For n = 60, the procedure is able to solve the K = 2 instance for f_2 and the K = 2, 3 and 4 instances for f_3 within the 2-hour limit. No $n \geq 90$ instances could be solved for either f_2 or f_3 within the specified time limit.

	Objective function value			CPU time in seconds		
K	f_1	f_2	f_3	f_1	f_2	f_3
2	63	6292	403.333	0.120	0.04	0.02
3	52	3063	304.456	0.160	5.41	0.06
4	43	2045	259.069	0.130	865.8	3.76
5	37	--	220.065	0.150	> 2 hours	24.55
6	28	--	189.375	0.160	> 2 hours	194.81
7	27	--	163.875	0.150	> 2 hours	1173.17
8	23	--	--	0.130	> 2 hours	> 2 hours

Table 3 Branch-and-bound results for $n = 30$

We now examine our choice for representing solutions within the search process. A "natural" representation consists of an n-dimensional vector $x = (x_1, \ldots, x_n)$, where x_i ($0 \le x_i \le K$) is the index of the cluster to which object i is assigned. Since clusters have no unique identity, this representation has the disadvantage of producing a potentially large number of x-vectors that map to the same clustering configuration. As indicated in section 3.2, we chose a representation in which K objects become the centroids of the clusters in a given solution. This representation attempts to minimize the "symmetry" problems cause by the lack of cluster identity. However, there is no guarantee that the optimal clustering (with respect the criteria under consideration) can be represented with centroid mechanism that we described above.

The purpose of our next experiment is to evaluate the merit of the centroid-based solution representation. We compare the performance of two methods, one (referred to as *Naïve*) that consists of randomly assigning objects to clusters and another (referred to as *Centroids*) that randomly selects K objects to be the centroids of the clusters and where the remaining n-K objects are assigned to the closest cluster (as determined by the dissimilarity between the object under consideration and the cluster

centroid). We generate 100*n* solutions with each method for each instance. Table 4 reports for both methods, on each group of seven instances (K = 2, ..., 8), the average ± the standard deviation of the solution values for each objective function. The last row shows the average of the average values and standard deviations over the 35 instances.

	Naïve				*Centroids*			
n	f_1	f_2	f_3	f_4	f_1	f_2	f_3	f_4
30	95.59 ± 35.57	4458.38 ± 4321.00	522.63 ± 160.67	200.20 ± 152.70	57.58 ± 55.86	3299.98 ± 7742.71	294.58 ± 198.81	91.12 ± 78.71
60	110.77 ± 28.00	19162.73 ± 9271.57	1213.25 ± 206.91	220.33 ± 77.31	70.86 ± 60.57	14840.47 ± 35201.86	672.10 ± 437.92	97.89 ± 81.09
90	111.13 ± 22.00	43832.46 ± 13533.00	1895.40 ± 218.23	222.99 ± 50.60	73.28 ± 61.29	33835.37 ± 85967.57	1042.04 ± 679.36	99.11 ± 69.69
120	120.53 ± 22.43	82076.26 ± 19208.71	2693.26 ± 228.59	234.19 ± 29.05	78.67 ± 65.00	62844.41 ± 168649.29	1470.51 ± 986.93	103.20 ± 70.64
150	120.66 ± 20.71	128755.13 ± 22400.86	3403.77 ± 234.64	234.74 ± 25.01	80.00 ± 66.14	99016.53 ± 254334.29	1863.16 ± 1204.17	103.82 ± 67.33
Average	111.74 ± 25.74	55656.99 ± 13747.03	1945.66 ± 209.81	222.49 ± 66.93	72.08 ± 61.77	42767.35 ± 110379.14	1068.48 ± 701.44	99.03 ± 73.49

Table 4 Random sampling results

The results in Table 4 reveal the advantage of basing a search procedure in a solution representation that uses centroids. For each problem size and optimization criterion, we find significant differences between the average quality of the solutions in the random samples. This justifies our choice of the centroids method for representing solutions in our search procedure.

We have verified that the solution representation based on centroids is an effective solution method in its own right when tackling small problem instances. Specifically, we applied the *Centroids* random sampling procedure to the problems with n = 30 associated with Table 3 and, in computing times

not exceeding an average of 5 seconds, the procedure is able to match all the known optimal solutions to these problems. The quality of the solutions found with *Centroids* deteriorates (when compared to the solutions found with our metaheuristic) as the size of the problem instances increases.

In the next subsections, we present the results of applying our metaheuristic to multiobjective problems. We used the following parameter settings:

$b = 2*p$

$NSize = 2*K$

$TabuTenure = 20$

$InitPhase = 3$

Our solution procedure is capable of tackling problems that simultaneously consider multiple criteria and multiple matrices. We divide our main experiments with synthetic data into three blocks: experiments with multiple criteria, experiments with multiple matrices and experiments with both multiple criteria and multiple matrices. In order to assess the quality of the output produce by multiobjective procedures, we consider three well known measures taken from the literature:

Number of points: This refers to the ability of finding efficient points. The preference is to find more rather than fewer (potentially) efficient points.

SSC: This metric, suggested by Ziztler and Thiele (1999) measures the *Size of the Space Covered* (SSC). SSC measures the volume of the dominated points and, therefore, the larger the value the better.

$C(A,B)$: This is known as the coverage of two sets measure and was also suggested by Zizler and Thiele (1999). $C(A,B)$ represents the proportion of points in the estimated efficient frontier B that are dominated by the efficient points in the estimated frontier A.

4.1 Experiments with Multiple Criteria

For this set of experiments, we employ the procedure (which we will refer to as *Bi-Heur*) recently published by Brusco and Cradit (2005). There are two main differences between our metaheuristic (SSPMO) and *Bi-Heur*:

1. *Bi-Heur* considers f_1 as the primary objective and f_i (i = 2, 3 or 4) as the secondary objective while SSPMO simultaneously handles f_1, f_2, f_3 and f_4.

2. A single run of SSPMO yields an approximation of the efficient frontier while a single run of *Bi-Heur* results in a single (possibly) efficient point.

Hence, in order to compare our results, we executed SSPMO once for each problem instance in our data set, with n = 30 to 150 and K = 2 to 8. For each run, we record the approximation of the efficient frontier and the total execution time. We then used *Bi-Heur* to find solutions to the bicriterion problems (f_1, f_2), (f_1, f_3) and (f_1, f_4). For each bicriterion problem, *Bi-Heur* was executed multiple times from randomly generated initial partitions (as recommended by the authors). The number of runs was limited by the amount of time used during the SSPMO run. In other words, we launched *Bi-Heur* from a random partition as many times as possible as long as the total computational time did not exceed the one employed by SSPMO. We rotated through the 3 bi-criterion problems in order to allocate similar computational time to all of them.

Table 5 reports the average of the objective function values found by each approach. The accompanying Table 6 shows two multiobjective measures: average SSC and average number of points.

In addition, Table 6 includes the average computational effort (in CPU seconds) associated with obtaining each (potentially) efficient point. We report this because the total computational effort is the same for both approaches.

	Bi-Heur				SSPMO			
n	f_1	f_2	f_3	f_4	f_1	f_2	f_3	f_4
30	42.43	2198.86	247.97	63.08	39.00	2156.14	240.54	47.24
60	54.43	9843.14	566.01	69.93	47.71	9688.14	546.89	54.43
90	54.29	22373.29	872.89	71.03	49.29	22367.00	865.27	58.21
120	59.57	42665.71	1245.97	70.19	53.71	42272.00	1223.17	59.66
150	58.14	67063.57	1581.84	70.59	54.86	67014.71	1560.95	61.73
Avg.	53.77	28828.91	902.93	68.96	48.91	28699.60	887.36	56.25

Table 5 Average objective function values

	Bi-Heur			SSPMO		
n	*No. of points*	*SSC*	*Time*	*No. of points*	*SSC*	*Time*
30	3.00	0.23	2.80	83.86	0.51	4.20
60	3.00	0.20	31.03	150.14	0.52	23.48
90	3.00	0.13	132.78	224.43	0.46	156.12
120	3.00	0.16	424.67	248.86	0.44	183.35
150	3.00	0.12	1312.78	287.29	0.45	411.55
Avg.	3.00	0.17	380.81	198.91	0.48	155.74

Table 6 Comparison of SSPMO and *Bi-Heur* using multiobjective measures

Bi-Heur is very sensitive to the initial partition and this is why its authors recommend executing the heuristic from 100 initial random partitions. This results in a very time-consuming process that is not

capable of generating more than one (possibly) efficient solution for each bi-criterion problem (as shown in the second column of Table 6). Tables 5 and 6 show the advantage of our procedure when tackling clustering problems with multiple criteria. SSPMO produces approximations of the efficient frontier with SSC values that are at least twice as large as those produced by *Bi-Heur*. The results are similar when comparing SSPMO and *Bi-Heur* taking each bicriterion problem at a time. In other words, SSPMO outperforms *Bi-Heur* when examining the results of the three bicriterion problems (f_1, f_2), (f_1, f_3) and (f_1, f_4). Using the output from this experiment, we also calculated the associated $C(A,B)$ values:

C(SSPMO, *Bi-Heur*) = 0.345

C(Bi-Heur, SSPMO) = 0.007

These values are additional evidence of the merit of SSPMO in the context of multiobjective clustering. The aggregate coverage of two sets indicates that almost 35% of the points estimated to be efficient by *Bi-Heur* are actually denominated by points found by SSPMO. The opposite only occurs less than 1% of the time.

4.2 Experiments with Multiple Dissimilarity Matrices

Brusco and Stahl (2005) discussed the application of branch-and-bound to the bicriterion problem of minimizing f_3 when two dissimilarity matrices (**A** and **B**) are considered. A single execution of their FORTRAN code (*bbbiwcss.for*) results in an efficient point. In order to obtain several efficient points, the code must be executed with different weight values. We modified the code as suggested in Brusco and Stahl (2005). In particular, we created a loop with a variable *iloop* varying from 0 to 10. The weight for matrix **A** is *iloop*/10 while the weight for matrix **B** is (1 – *iloop*/10). In this way, 11 efficient points are obtained, including the optimal solutions associated with each matrix.

Brusco's and Stahl's exact procedure is capable of tackling problems with up to 30 objects. We applied the modified branch-and-bound code to our two 30-object matrices and set $K = 2, \ldots, 8$. The

branch-and-bound code generated only two distinct efficient points for problems with $K > 3$. For the problem with $K = 2$, only 3 efficient points were found. The approach required 221 seconds to find 6 efficient points for the problem with $K = 6$. SSPMO required 9 seconds to find an approximation of the efficient frontier with 86 points, including 5 of the efficient points found by the branch-and-bound procedure. Figure 1 shows SSPMO's approximate frontier and the 6 efficient points.

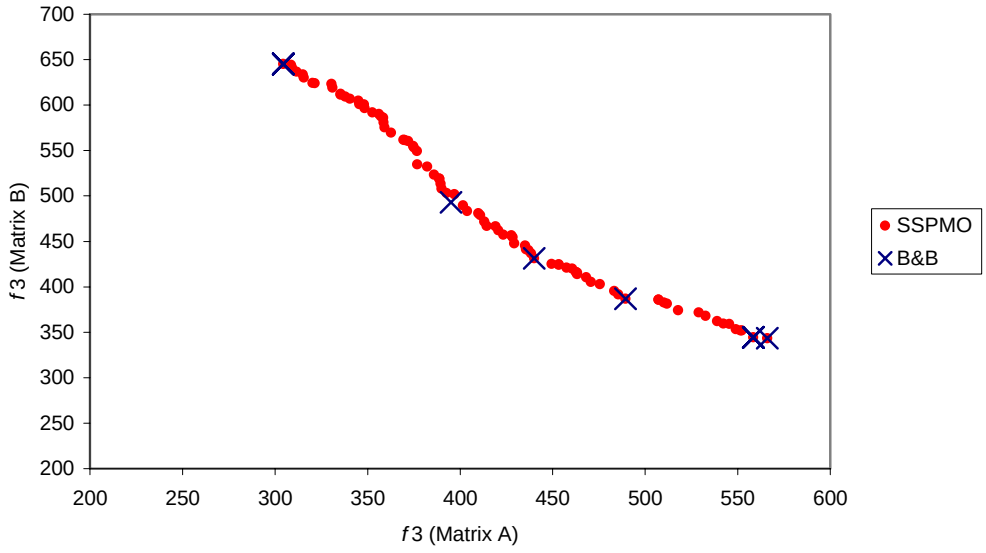

Figure 1 Efficient frontier for the minimization of f_3 with two 30-object matrices and $K = 3$

The only efficient point that SSPMO is not able to find is (395.111, 492.972). The best approximations to this point are (401.627, 489.404) and (396.942, 501.706). In both cases, neither objective function value is more than 2% away from the efficient point. We do not present the results for $K = 3$ because in all of these instances SSPMO matches the known efficient solutions in a fraction of the time required by the branch-and-bound code.

As part the experimentation with multiple matrices, we ran SSPMO using the two 30-object matrices, $K = 2$, 3 and 4 and all the objective functions (f_1 to f_4) considered at the same time. To the best of our knowledge, there is no published procedure that is capable of tackling such a problem. The results

that we present in Table 7 compare the performance of SSPMO and the simple *Naïve* method described above. The rationale for the experiment is twofold. We would like to determine whether SSPMO provides an advantage over a simple approach and we would like to establish a benchmark for future research in this area.

	Naïve			SSPMO		
K	*No. of points*	*SSC*	*Time*	*No. of points*	*SSC*	*Time*
2	473	0.016538	1217.5	1095	0.017594	1183.5
3	2847	0.096896	9228.2	8496	0.103708	9251.3
4	8627	0.070108	7011.3	15496	0.098749	8007.2

Table 7 Results for multiple matrices ($n = 30$) and multiple objective functions ($f_1, \ldots, f_4$).

Table 8 shows the coverage of two sets measure associated with the results in Table 7

K	*C*(SSPMO, Naïve)	*C*(Naïve, SSPMO)
2	0.116	0.003
3	0.206	0.002
4	0.217	0.017

Table 8 Coverage of two sets measure for the results reported in Table 7

Table 7 and 8 reveal the usefulness of our approach. Although the absolute differences in *SSC* values may seem small, the relative differences are indeed significant. When solving problems with multiple matrices and objective functions, *SSC* values are not expected to be much larger than zero. This is due to the increase in dimensionality of the objective function space. As the number of dimensions increases, the number of efficient points in the approximation of the frontier must increase exponentially in order to maintain a high *SSC* value. The volume calculation (i.e., *SSC*) in Table 7 shows the effect of dealing with

8 objective functions at the same time (i.e., 4 different criteria and 2 sources of data). This experiment shows that current metaheuristic technology is not capable of dealing with the exponential increase in the number of points that is necessary to obtain better volume measures in high-dimensional multiobjective combinatorial optimization problems. The merit of SSPMO is therefore more obvious when examining the $C(A,B)$ values reported in Table 8.

4.3 Experiments with Segmentation

We perform one final experiment in which we use real data from a market segmentation problem. Given a set of customers, a set of descriptor variables and a response variable, the problem is to find a clustering of the customers that maximizes both a measure of homogeneity and the explanation power of the response variable. As described in Brusco, Cradit and Stahl (2002), the bicriterion problem consists of finding partitions that simultaneously maximize the proportion of variation explained by the descriptor variables and the response variable. The "classical" market segmentation problem deals with creating homogenous groups and therefore deals with the single objective of maximizing the variation explained by the descriptor variables. In the bicriterion problem, the maximization of the variation explained by the response variable is also considered.

The input for this experiment consists of market data (variables and observations). The objective functions are $\omega^2(C)$ and $\eta^2(C)$, as denoted in Brusco, Cradit and Stahl (2002). Both functions represent the ratio of the between-clusters sum of squares and the total sum of square, for a given cluster C. The sum of squares in $\omega^2(C)$ are calculated using the descriptor variables while the sum of squares in $\eta^2(C)$ are calculated using the response variable. In other words, $\omega^2(C) = g_3^l$ (see section 1.2) when the dissimilarity matrix is obtained by calculating the Euclidean distances between every pair of observations (customers) using the descriptor variables. The same is true for $\eta^2(C)$ when using the response variable. The simulated annealing procedure (SAH) of Brusco, Cradit and Stahl (2002) operates in the scalar

objective function that assigns a weight to $\omega^2(C)$ and another weight to $\eta^2(C)$. The sum of the weights is required to be equal to one.

As discussed in Ehrgott and Gandibleux (2000), the biggest additional challenge when solving multiobjective combinatorial optimization problems as opposed to multiobjective linear programs is the presence of efficient solutions that are not optimal for any scalarization of the problem. Scalarization refers to the transformation of the multiobjective problem into a single-objective problem by means of optimizing a function that is the weighted sum of the multiple objective functions (as discussed in section 2).

Efficient solutions that cannot be found by solving the scalar optimization problem are referred to as unsupported efficient solutions. Those efficient solutions that are optimal for the scalar problem are called supported efficient solutions. The set of unsupported efficient solutions is important and it has been shown that the ratio of unsupported to supported points is large in multiobjective combinatorial optimization problems. More details about searching for unsupported efficient solutions can be found in Ulungu and Teghem (1994) or in Visée et al. (1998)

Lichtenstein, Burton and Netemeyer (1997) tackle a market segmentation problem related to consumers' proneness to deals. In their study, they surveyed customers walking out of two grocery stores. The survey focused on measuring eight deal-specific proneness types: proneness toward coupons, sales, cents-off, buy-one-get-one-free, free-gift-with-purchase, end-of-aisle displays, rebates/refunds, and contests/sweepstakes. They used SPSS to find a *K*-means solution with 2 and 3 clusters. The resulting clusters were used to perform an ANOVA on ten dependent variables. The clusters obtained following this method are meant to be homogenous with respect to the independent variables (i.e., with respect to the eight deal-specific proneness types).

We used this market data to compare the performance of SSPMO and SAH in the context of the bicriterion market segmentation problem. Because SAH can deal only with one dependent variable, we chose "quantity of coupons redeem", which is one of the ten response variables examined by Lichtenstein, Burton and Netemeyer (1997). Hence, the data set consists of 524 observations (customers),

8 independent variables and 1 dependent variable. Figure 2 shows the approximations of the efficient frontier found by SAH and SSPMO for $K = 3$.

SSPMO finds 748 points and in the same amount of computational time (3646.01 CPU seconds) SAH finds only 11 points. Figure 2 shows that the SAH is not capable of finding efficient points in areas of the frontier that are not convex. These are the areas where unsupported efficient points lie and where scalar-based approaches fail. The results using other independent variables in the data set are the same. Results with $K = 2$ show an even more dramatic picture of the inability of SAH to find efficient points because the entire efficient frontier is non-convex.

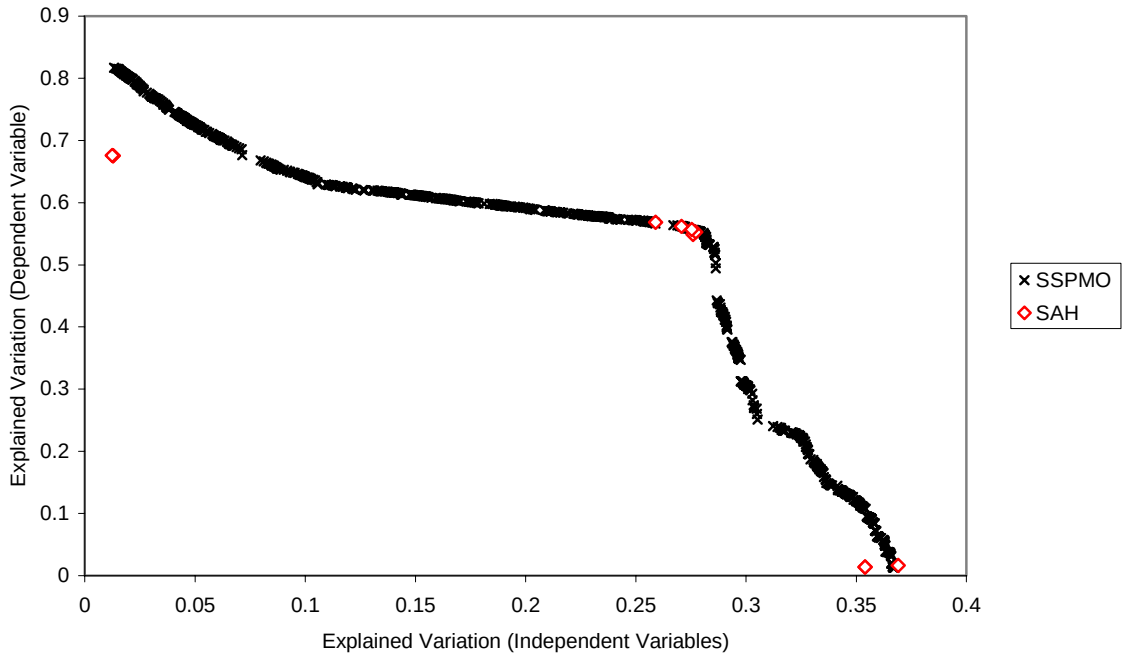

Figure 2 Efficient frontier approximation for market segmentation problem with $K = 3$

The extension of SAH described by Brusco, Cradit and Tashchian (2003) attempts to find efficient solutions to market segmentation problems that consider more than one response variable. We did not

conduct experiments with this modified version of SAH because we expect to encounter similar problems regarding the existence of unsupported efficient points.

5 Conclusions

We have described the development and testing of a metaheuristic approach to multiobjective clustering problems. The proposed procedure is based on two well known metaheuristic methodologies: tabu search and scatter search. We have been able to show the merit of our procedure by applying it to several clustering problems and comparing it to exiting heuristic and exact approaches. We believe that this study advances the state-of-the-art in finding good approximations of efficient frontiers for multiobjective clustering problems. Our experimentation goes beyond problems related to problems with artificially created dissimilarity matrices and shows that the proposed approach is effective and can deal with practical problems, such as those arising in market segmentation. Also, the structure of our search procedure is such that it is not affected by the presence of unsupported efficient points. Multiobjective problems in data analysis seem to be a fertile application ground for metaheuristic optimization methods. We expect to see additional development of multiobjective metaheuristic procedures that target problems in this field.

Acknowledgments

We wish to thank Prof. Michael Brusco for sharing his branch-and-bound and SAH computer programs. We also like to thank Prof. Donald Lichtensein for sharing his market data. The generosity of colleagues like Prof. Brusco and Lichtenstein is what allows researchers to advance knowledge in our fields of study.

6 References

Brusco, M. J. and J. D. Cradit (2005) "Bicriterion Methods for Partitioning Dissimilarity Matrices," *British Journal of Mathematical and Statistical Psychology*, vol. 58, no. 2, pp. 319–332.

Brusco, M. J., J. D. Cradit and S. Stahl (2002) "A Simulated Annealing Heuristic for a Bicriterion Partitioning Problem in Market Segmentation," *Journal of Marketing Research,* vol. XXXIX, pp. 99-109.

Brusco, M. J., J. D. Cradit and A. Tashchian (2003) "Multicriterion Custerwise Regression for Join Segmentation Settings: An Application to Customer Value," *Journal of Marketing Research,* vol. XL, pp. 225-234.

Brusco, M. J. and S. Stahl (2005) *Branch and Bound Applications of Combinatorial Data Analysis,* Springer: New York, ISBN 0-387-25037-9.

Delattre, M. and P. Hansen (1980) "Bicriterion Cluster Analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 2, no. 4, pp. 227-291.

Erhgott, M. and X. Gandibleux (2000) "A Survey and Annotated Bibliography on Multiobjective Combinatorial Optimization," *OR Spektrum*, vol. 22, pp. 425-460.

Glover, F. and M. Laguna (1997) *Tabu Search,* Kluwer Academic Publishers: Boston, MA.

Hansen, P. and M. Delattre (1978) "Complete-Link Cluster Analysis by Graph Coloring," *Journal of the American Statistical Association,* vol. 73, no. 362, pp. 397-403.

Hansen, P., B. Jaumard and O. Frank (1989) "Maximum Sum-of-Splits Clustering," *Journal of Classification,* vol. 6, pp. 177-193.

Krieger, A. M. and P. E. Green (1996) "Modifying Cluster-Based Segments to Enhance Agreement with an Exogenous Response Variable," *Journal of Marketing Research,* vol. 33, pp. 351-363.

Laguna, M. and R. Martí (2003) *Scatter Search: Methodology and Implementations in C,* Kluwer, Boston, MA.

Lichtenstein, D. R., S. Burton and R. G. Netemeyer (1997) "An Examination of Deal Proness Across Sales Promotion Types: A Consumer Segmentation Perspective," *Journal of Retailing,* vol. 73, no. 2, pp. 283-297.

Molina, J., M. Laguna, R. Martí and R. Caballero (2006) "SSPMO: A Scatter Tabu Search Procedure for Non-Linear Multiobjective Optimization," to appear in *INFORMS Journal on Computing*.

Ulungu, E. and J. Teghem (1994) "The Two-phase Method: An efficient Procedure to Solve Bi-objective Combinatorial Optimization Problems," *Foundations of Computing and Decision Sciences*, vol. 20, no. 2, pp. 149-165.

Visée, M., J. Teghem, M. Pirlot, and E. Ulungu (1998) "Two-Phase method and Branch and Bound Procedures to Solve the Bi-objective Knapsack Problem," *Journal of Global Optimization*, vol. 12, pp. 139-155.

Zeleny, M. (1982) *Multicriteria Decision Making,* McGraw-Hill: New York.

Zitzler, E. and L. Thiele (1999) “Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach,” *IEEE Transactions on Evolutionary Computation,* vol. 3, pp. 257–271.