PSYCHOMETRIKA
2007
DOI: 10.1007/s11336-007-9013-4

A COMPARISON OF HEURISTIC PROCEDURES FOR MINIMUM WITHIN-CLUSTER SUMS OF SQUARES PARTITIONING

MICHAEL J. BRUSCO

FLORIDA STATE UNIVERSITY

DOUGLAS STEINLEY

UNIVERSITY OF MISSOURI-COLUMBIA

Perhaps the most common criterion for partitioning a data set is the minimization of the within-cluster sums of squared deviation from cluster centroids. Although optimal solution procedures for within-cluster sums of squares (WCSS) partitioning are computationally feasible for small data sets, heuristic procedures are required for most practical applications in the behavioral sciences. We compared the performances of nine prominent heuristic procedures for WCSS partitioning across 324 simulated data sets representative of a broad spectrum of test conditions. Performance comparisons focused on both percentage deviation from the "best-found" WCSS values, as well as recovery of true cluster structure. A real-coded genetic algorithm and variable neighborhood search heuristic were the most effective methods; however, a straightforward two-stage heuristic algorithm, HK-means, also yielded exceptional performance. A follow-up experiment using 13 empirical data sets from the clustering literature generally supported the results of the experiment using simulated data. Our findings have important implications for behavioral science researchers, whose theoretical conclusions could be adversely affected by poor algorithmic performances.

Key words: combinatorial data analysis, cluster analysis, heuristics, sum of squares criterion.

1. Introduction

Monte Carlo comparisons in the cluster analysis literature are especially prevalent because of the large number of methodological alternatives and implementation decisions faced by analysts. Reviews of many of these comparisons are offered by Arabie and Hubert (1992, 1996) and Steinley (2006a). In this paper we focus on a Monte Carlo comparison of methods for one of the most frequently selected criteria in the clustering literature: minimizing the within-cluster sums of squares (WCSS).

We consider a collection of N objects contained in the set $S = \{o_1, o_2, \ldots, o_N\}$ with the corresponding index set $C = \{1, 2, \ldots, N\}$. Each object is measured on V variables and the data are contained in the $N \times V$ matrix, $\mathbf{X} = [x_{iv}]$, whose elements contain the measure of variable v on object o_i (for all $1 \leq i \leq N$ and $1 \leq v \leq V$). We assume that the desired number of clusters, K, is prespecified, and the number of feasible partitions of the N objects into K clusters can, therefore, be computed as a Stirling number of the second kind. One possible formula for the number of feasible partitions is offered by Hand (1981):

$$\frac{1}{K!}\sum_{k=0}^{K}(-1)^k \binom{K}{k}(K-k)^N. \tag{1}$$

Requests for reprints should be sent to Michael J. Brusco, Department of Marketing, College of Business, Florida State University, Tallahassee, FL 32306-1110, USA. E-mail: mbrusco@cob.fsu.edu

The set containing all feasible K-cluster partitions is denoted Π_K, and members of this set are partitions $\pi_K = \{S_1, S_2, \ldots, S_K\}$. Each set S_k $(1 \leq k \leq K)$ contains the objects assigned to cluster k of the partition, π_K. The set C_k $(1 \leq k \leq K)$ containing the indices of the objects in S_k has a one-to-one relationship with S_k. Further, the cardinality of S_k (and C_k) represents the number of objects assigned to cluster k and is denoted N_k $(1 \leq k \leq K)$ and $\sum_{k=1}^{K} N_k = N$.

Although other distance metrics are possible, we consider the WCSS for a partition, π_K, based on squared Euclidean distance, which is computed as follows:

$$WCSS(\pi_K) = \sum_{k=1}^{K} \sum_{i \in C_k} \sum_{v=1}^{V} \left(x_{iv} - \bar{x}_{vk}\right)^2, \tag{2}$$

where $\bar{x}_{vk} = \sum_{i \in C_k} x_{iv}/N_k$ is the mean of variable v in cluster k (for all $1 \leq v \leq V$ and $1 \leq k \leq K$). The objective criterion of the minimum within cluster sum of squares partitioning problem (MWCSSP) is

$$\text{Minimize:} \quad \left[WCSS(\pi_K) \mid \pi_K \in \Pi_K\right]. \tag{3}$$

The MWCSSP is known to be NP-hard (Brucker, 1978; Day, 1996). Some recent progress has been made for producing guaranteed optimal solutions for problems of nontrivial size. Dynamic programming methods can obtain optimal solutions for small problems with up to about $N = 30$ objects (Hubert, Arabie, & Meulman, 2001, Chap. 4; van Os & Meulman, 2004). A branch-and-bound algorithm for MWCSSP was originally proposed by Koontz, Narendra, and Fukunaga (1975), and later refined by Diehr (1985) and Brusco (2006). Du Merle, Hansen, Jaumard and Mladenoviĉ (2000) developed an effective procedure that incorporates an interior point algorithm, column generation, a neighborhood search heuristic, and branch-and-bound methods. Despite the progress in the development of optimal procedures for MWCSSP, heuristic methods are typically required for most practical applications.

The earliest heuristic procedures for MWCSSP were popularized in the 1960s under the constraint of limited computing power (Forgy, 1965; Jancey, 1966; MacQueen, 1967). These procedures typically alternate between computation of the cluster centroids and reassignment of objects, with a few minor differences that will be highlighted in the next section. Somewhat more computationally demanding local search operations were developed in the 1970s (Banfield & Bassil, 1977; Hartigan & Wong, 1979), including relocation of objects and pairwise interchange of objects in different clusters. More recently, heuristic developments have focused on metaheuristics capable of escaping poor local minima. These methods include simulated annealing (Babu & Murty, 1994; Klein & Dubes, 1989; Selim & Al-Sultan, 1991; Sun, Xie, Song, Wang, & Yu, 1994a; Sun, Xu, Liang, Xie, & Yu, 1994b), tabu search (Al-Sultan, 1995; Pacheco & Valencia, 2003; Sung & Jin, 2000), genetic algorithms (Babu & Murty, 1993; Jones & Beltramo, 1991; Krishna & Murty, 1999; Maulik & Bandyopadhyay, 2000; Pacheco & Valencia, 2003), and variable neighborhood search (Hansen & Mladenoviĉ, 2001).

Despite the prevalence of new heuristic methods for MWCSSP, there has been little systematic comparison of methods across a broad range of data conditions. Recent comparative analyses have been restricted to only a few empirical data sets that are measured in a low-dimensional space (Hansen & Mladenoviĉ, 2001; Pacheco & Valencia, 2003). Therefore, one contribution of this current manuscript is a comparative analysis of prominent heuristic procedures for MWCSSP across a broad range of data sets representative of those that are typical for the social sciences. Although our comparative analyses focus primarily on the ability of existing methods to minimize WCSS, we also recognize that the recovery of some known cluster structure has often been of paramount importance in the quantitative behavioral sciences literature (Brusco, 2004; Brusco & Cradit, 2001; Dimitriadou, Dolniĉar, & Weingessel, 2002; Milligan, 1980a, 1985, 1989; Milligan & Cooper, 1986, 1988; Steinley, 2003, 2006b). Therefore, we consider the cluster structure recovery performances of the algorithms, as well as their ability to minimize the WCSS criterion.

In the next section of this paper we provide a thorough description of the heuristic procedures evaluated in our experimental study. Subsequent sections describe the experimental test conditions and the results of the computational analyses. The paper concludes with a discussion of the implications of the findings, as well as recommendations for selecting algorithmic procedures.

2. Heuristic Solution Procedures

We selected nine heuristic procedures for comparison in our analyses. We endeavored to maintain consistency with previous implementations to the greatest extent possible.

2.1. *H-Means* (Forgy, 1965)

One of the earliest heuristic procedures for MWCSSP was suggested by Forgy (1965). We adopt Hansen and Mladenoviĉ's (2001) terminology and refer to the algorithm as H-means. A description of the algorithm is as follows:

Step 0. Read in an initial K-cluster partition, π_K.
Step 1. Compute $\bar{x}_{vk} = \sum_{i \in C_k} x_{iv}/N_k$ for all $1 \le v \le V$ and $1 \le k \le K$,
and $d_k^i = \sum_{v=1}^{V} (x_{iv} - \bar{x}_{vk})^2$ for $1 \le k \le K$ and $1 \le i \le N$.
Step 2. Obtain π'_K as follows: $[o_i \in S_k$ and $i \in C_k \mid d_k^i = \min_{(1 \le l \le K)}(d_l^i)]$.
Step 3. If $\pi'_K = \pi_K$, then Stop. Otherwise, return to Step 1.

One of the limitations of H-means is that it can potentially terminate with one or more empty clusters. One remedy for this degeneracy problem is to reassign objects most distant from their cluster centroids to the empty clusters. For consistency with Hansen and Mladenoviĉ (2001), we refer to the resulting modified version of H-means as H-means+.

2.2. *K-Means* (Jancey, 1966; MacQueen, 1967)

The K-means algorithm systematically assesses all possible reassignments of objects to a different cluster until no further reassignment improves $WCSS(\pi_K)$. The key component of this process is the computation of the change in the WCSS value that results from moving object i from its current cluster k to a new cluster ($l \ne k$). This change is efficiently computed as

$$\Delta_{kl}^i = \frac{N_l}{(N_l + 1)} \sum_{v=1}^{V} \left(\bar{x}_{vl} - x_{iv}\right)^2 - \frac{N_k}{(N_k - 1)} \sum_{v=1}^{V} \left(\bar{x}_{vk} - x_{iv}\right)^2. \tag{4}$$

With this definition in place, the K-means algorithm is described as follows:

Step 0. Set $\phi = 0$ and read in an initial K-cluster partition, π_K.
Step 1. Perform systematic assessment of all single-object cluster reassignments:
For $i = 1$ to N
For $l \ne (k \mid i \in C_k$ and $|C_k| > 1)$
compute Δ_{kl}^i
if $\Delta_{kl}^i < 0$, then
move o_i from S_k to S_l, set $\phi = 1$.
end if
next l
next i
Step 2. If $\phi = 0$, then Stop. Otherwise, set $\phi = 0$ and go to Step 1.

2.3. *HK-Means* (Hansen & Mladenoviĉ, 2001)

A solution that is locally optimal for the H-means+ criterion could possibly be improved by applying K-means, but the reverse is not true. Based on this property, Hansen and Mladenoviĉ (2001) suggested a two-phase solution strategy using H-means+ and K-means, successively. The procedure is concisely stated as follows:

Step 0. Read in an initial K-cluster partition, π_K.
Step 1. Perform H-means+.
Step 2. Perform K-means.

2.4. *KI-Means* (Banfield & Bassil, 1977)

Banfield and Bassil (1977) provided a two-stage heuristic procedure, which uses the K-means algorithm in the first stage, and a pairwise interchange heuristic (which we will call "I-Means") in the second stage. These two stages are iteratively implemented until neither heuristic operation can further improve WCSS. A formal description of the algorithm is as follows:

Step 0. Read in an initial K-cluster partition, π_K.
Step 1. Perform K-means.
Step 2. Perform I-Means.
  Step 2a. Set $\phi = 0$.
  Step 2b. Perform systematic assessment of all pairwise interchanges:
    For $(i < j) \in C \mid i \in C_k$, $j \in C_l$ and $l \neq k$
      compute δ_{kl}^{ij}, which represents the effect on WCSS from the pairwise interchange made by moving o_i from S_k to S_l and o_j from S_l to S_k
      if $\delta_{kl}^{ij} < 0$, then
        move o_i from S_k to S_l, and move o_j from S_l to S_k, and set $\phi = 1$.
        end if
      next l
    next i
  Step 2c. If $\phi = 0$, then Stop. Otherwise, set $\phi = 0$ and go to Step 2b.
Step 3. If any objects changed cluster membership during the most recent execution of Step 1 or Step 2, then return to Step 1; otherwise, Stop.

2.5. *J-Means*+ (Hansen & Mladenoviĉ, 2001)

Hansen and Mladenoviĉ (2001) developed the J-Means+ heuristic procedure that allows for a broader search of the solution neighborhood by considering the reassignment of an entire centroid rather than just a reassignment of an object. To present the J-means+ algorithm, we define $\mathbf{x}_i$ as a $1 \times V$ vector of measurements for row i in matrix $\mathbf{X}$. The algorithm is as follows:

Step 0. Read in an initial K-cluster partition, π_K, and let $\pi_K^* = \pi_K$ and $\pi_K'' = \pi_K$.
Step 1. Set $\Omega = \{i \mid \mathbf{x}_i \neq [\bar{x}_{1k}, \bar{x}_{2k}, \ldots, \bar{x}_{Vk}]$ for any $k\}$.
Step 2. Centroid replacement procedure
  For $i \in \Omega$
    For $k = 1$ to K
      set $[\bar{x}_{1k}, \bar{x}_{2k}, \ldots, \bar{x}_{Vk}] = \mathbf{x}_i$
      reassign all objects to their nearest cluster to create π_K'
      if $WCSS(\pi_K') < WCSS(\pi_K'')$ then
        set $WCSS(\pi_K'') = WCSS(\pi_K')$ and $\pi_K'' = \pi_K'$
      end if
    next k
  next i

Step 3. Run HK-means using π''_K as a starting solution and let π'''_K be the obtained solution.
Step 4. If $WCSS(\pi'''_K) < WCSS(\pi^*_K)$, then set $\pi^*_K = \pi'''_K$ and return to Step 1; otherwise, Stop.

2.6. *Tabu Search* (Pacheco & Valencia, 2003)

Tabu search is a general heuristic paradigm for combinatorial optimization problems that facilitates escape from local optima (Glover, 1989, 1990; Glover & Laguna, 1993; Glover, Taillard, & Werra, 1993). Escape is accomplished by a process that permits acceptance of suboptimal solutions, and forbids moves on a "tabu list" that would tend to yield an immediate return to the recently found local optimum. We selected Pacheco and Valencia's (2003) tabu search partitioning algorithm for evaluation because of its conceptual straightforwardness. Their algorithm defines $tabu\_tenure = K$ as the number of iterations that an object is forbidden to return to a cluster. The parameter $max\_noimprovement = 50$ represents the maximum number of iterations for which no improvement was realized, and defines the termination criterion for the algorithm. The $N \times K$ matrix, $\mathbf{Y} = [y_{ik}]$, contains the tabu record for the assignment of each object i to each cluster k (for $1 \le i \le N$ and $1 \le k \le K$). The constants *niter* and *iter_better* represent the current number of iterations of the algorithm and the last iteration that produced a solution with an improved objective value, respectively. The algorithm is as follows:

Step 0. Read in an initial K-cluster partition, π_K, and let $\pi^*_K = \pi_K$. Set niter $= 0$, $iter\_better = 0$, $max\_noimprovement = 50$; and $y_{ik} = -tabu\_tenure$ (for $1 \le i \le N$ and $1 \le k \le K$).
Step 1. Set $niter = niter + 1$.
Step 2. Compute Δ^i_{kl} as in (4) for all $1 \le i \le N$ and for $l \ne (k \mid i \in C_k$ and $|C_k| > 1)$.
Step 3. Choose i^*, k^*, l^* such that $\Delta^{i^*}_{k^*l^*} = (\min(\Delta^i_{kl}) \mid niter > y(i,l) + tabu\_tenure)$.
Step 4. Update π_K by assigning moving o_i from S_k to S_l and set $y_{ik} = niter$.
Step 5. If $WCSS(\pi_K) < WCSS(\pi^*_K)$, then set $\pi^*_K = \pi_K$ and $iter\_better = niter$.
Step 6. If $niter - iter\_better > max\_noimprovement$, then Stop; otherwise go to Step 1.

2.7. *Simulated Annealing* (Klein & Dubes, 1989)

The adaptation of the metallurgical process of annealing (Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller, 1953) for the traveling salesman combinatorial optimization problem was independently offered by Kirkpatrick, Gelatt, and Vecchi (1983) and Cerny (1985). Our implementation of simulated annealing for WCSS partitioning is most consistent with that of Klein and Dubes (1989) and uses single-object relocations to produce neighboring solutions. To establish the *temperature* for the algorithm, which controls the probability of accepting inferior solutions, we evaluate 5000 random relocations of objects and store the absolute minimum ($min\_change > 0$) and absolute maximum (max_change) in the WCSS. The initial temperature is set equal to max_change, and the min_change value designates a termination criterion. The algorithm is as follows:

Step 0. Read in an initial K-cluster partition, π_K, and let $\pi^*_K = \pi_K$. Set $cooling\_factor = .9$, $temperature\_length = 10N$, and perform relocation trials to obtain max_change and min_change. Compute the number of temperature reductions:

$$number\_of\_reductions = \left\lceil \frac{\log(\min\_change - \max\_change)}{\log(cooling\_factor)} \right\rceil,$$

where $\lceil \bullet \rceil$ is the smallest integer $\ge$ to "$\bullet$". Set $current\_temperature = max\_change$.

Step 1. Run Main Algorithm Loop
  For $nr = 1$ to *number_of_reductions*
    For $tl = 1$ to *temperature_length*
      randomly select an object o_i with $i \in C_k$ and $|C_k| > 1$
      randomly select a new cluster for o_i, $l \neq (k \mid i \in C_k)$
      compute Δ_{kl}^i
      if $\Delta_{kl}^i \leq 0$, then
        move o_i from S_k to S_l to update π_K
        if $WCSS(\pi'_K) < WCSS(\pi^*_K)$ then $\pi^*_K = \pi_K$
      else
        generate a uniformly distributed random number, *rn*
        let $b = -\Delta_{kl}^i / current\_temperature$
        if $rn < e^b$, then
          move o_i from S_k to S_l to update π_K
        end if
      end if
    next *tl*
    $current\_temperature = current\_temperature \times cooling\_factor$
  next *nr*.

2.8. *Genetic Algorithm* (Maulik & Bandyopadhyay, 2000)

Genetic algorithms are a class of stochastic optimization procedures that can be applied to a variety of problems (Belew & Booker, 1991; Forrest, 1993; Goldberg, 1989; Holland, 1975). The key components of a genetic algorithm are a population of *chromosomes* (or strings), which each possess a measure of fitness for the problem. Subsets of chromosomes are selected and subjected to operations that are analogous to the genetic characteristics of *crossover* and *mutation*. These operations create a new population of chromosomes, which is subsequently evaluated with respect to fitness, and this process is repeated for a fixed number of iterations. We selected the genetic algorithm for partitioning described by Maulik and Bandyopadhyay (2000), which uses a floating point representation for chromosomes (i.e., a real-coded genetic algorithm) that is convenient for identifying cluster centroids. Each chromosome is a string of KV elements that define cluster centroids. The algorithm is as follows:

Step 0. Set $population\_size = 100$, $crossover\_probability = .8$, and $mutation\_probability = .001$, $number\_of\_iterations = 100$, $current\_iteration = 0$, and $WCSS(\pi^*_K) = \infty$.
Step 1. Randomly create an initial population of solutions. Each chromosome is generated by randomly selecting K of the N objects and stringing together the V variable measurements of those objects to create a chromosome of length KV.
Step 2. Evaluation. For each chromosome in the population, reassign each of the N objects to its nearest chromosome (based on squared Euclidean distance) and update the chromosome elements (i.e., the centroids) after this assignment. Compute $WCSS(\pi_K)$ for each chromosome's partition and, if $WCSS(\pi_K) < WCSS(\pi^*_K)$ then set $\pi^*_K = \pi_K$.
Step 3. Termination. Set $current\_iteration = current\_iteration + 1$. If $current\_iteration > number\_of\_iterations$, then Stop; otherwise, go to Step 4.
Step 4. Fitness. Compute the fitness of each chromosome as $1/WCSS(\pi_K)$.
Step 5. Selection and Simple (One-Point) Crossover. Perform selection and crossover operations:
  For $ps = 1$ to $population\_size/2$.
    randomly select two chromosomes, ω and θ, using a roulette wheel strategy and biasing selection probability based on the fitness measures computed in Step 4.

generate a uniformly distributed random number, *rn*
if *rn* > *crossover_probability* then
add ω and θ to the new population pool
else
randomly select an integer, κ, on the interval [1, *KV*]
create the chromosome ω' from the first κ genes in ω and last $KV - \kappa$ genes in θ
create the chromosome θ' from the first κ genes in θ and last $KV - \kappa$ genes in ω
add ω' and θ' to the new population pool
end if
next *ps*.

Step 6. Mutation. Perform mutation operations using the following algorithm:
For *ps* = 1 to *population_size*.
For *kv* = 1 to *KV*
generate a uniformly distributed random number, *rn*
if *rn* < *mutation_probability*
randomly select a new value for the gene based on a uniform distribution
in a range that is defined by ±100% of the current value of the gene.
end
next *kv*
next *ps*
Return to Step 2.

2.9. *Variable Neighborhood Search* (Hansen & Mladenoviĉ, 2001)

Hansen and Mladenoviĉ (2001) proposed variable neighborhood search as an alternative metaheuristic for MWCSSP. A key feature of this approach is its systematic search of the solution space via increasingly larger neighborhood jumps. The key parameter is *neighbormax*, which represents the number of neighborhoods evaluated during the search process. For consistency with Hansen and Mladenoviĉ's implementation, we use *neighbormax* = 10. Following Hansen and Mladenoviĉ's implementation we assume an upper time limit, *time_max*, as a stopping rule. Based on the findings of Pacheco and Valencia (2003), we use HK-Means rather than J-Means in Step 3 of the algorithm, which is described as follows:

Step 0. Read in an initial K-cluster partition, π_K, and let $\pi_K^* = \pi_K$. Set *time_max* = some desired value and *neighbormax* = 10.
Step 1. Set *neighbor* = 1.
Step 2. Relocate *neighbor* randomly selected objects to randomly selected new clusters to create a new partition, π'_K.
Step 3. Apply HK-Means using π'_K as the starting solution. Let the returned partition be represented as π''_K.
Step 4. If $WCSS(\pi''_K) < WCSS(\pi^*_K)$, then set $\pi^*_K = \pi''_K$ and go to Step 1; otherwise set *neighbor* = *neighbor* + 1 and go to Step 5.
Step 5. If *neighbor* ≤ *neighbormax*, then go to Step 2; otherwise, go to Step 6.
Step 6. If elapsed time > *max_time*, then Stop; otherwise go to Step 1.

3. Computational Experiments for Simulated Data Sets

3.1. *Experimental Test Conditions*

We developed an experimental study to evaluate the nine heuristic procedures described in Section 2. To consider a broad range of conditions representative of data sets encountered in

the behavioral sciences, we varied the number of clusters, the number of variables, the relative density of each cluster, and the degree of overlap between clusters. For the number of clusters and variables, we examined values of $K = 4$, 6, and 8 and $V = 4$, 6, 8, and 10, respectively. To vary the relative density of each cluster (i.e., the number of objects within each cluster), the procedure popularized by Milligan (1980a) was implemented. Specifically, the following three conditions were considered:

(a) all clusters had an equal number of observations ($D = 1$)
(b) one cluster contained 10% of the total number of observations while the remaining 90% of observations were equally distributed among the other $K - 1$ clusters ($D = 2$); and
(c) one cluster contained 60% of the observations while the remaining 40% of observations were equally distributed among the other $K - 1$ clusters ($D = 3$)

Condition $D = 2$ focuses on the ability to be able to recover a smaller cluster in the presence of several larger clusters, whereas condition $D = 3$ focuses on finding several smaller clusters in the presence of one large cluster. The total number of observations across all clusters was $N = 200$ for each test problem.

The methodology for varying the degree of overlap between the clusters is slightly more complex. The first step is choosing the appropriate distributions from which to generate the clusters. Milligan (1980a) popularized the use of truncated multivariate normal distributions; however, Steinley (2006b) recently examined the performance of the H-means+ procedure when the cluster structure was generated from the following distributions: uniform distribution, triangular distribution, multivariate normal with equal variances, multivariate normal with unequal variances, and a mixed variable set where each of the variables was randomly chosen (with equal probability) to be generated from one of the four aforementioned distributions. It is well known (see Steinley, 2006a; 2006b) that the WCSS loss function performs best when the clusters arise from multivariate normal clusters with covariance matrices proportional to the identity (this is the equivalent to multivariate normal with equal variances condition). The uniform distribution simulates clusters that do not have a discernible "core" but have objects distributed evenly across the space within the cluster boundaries. The triangular distribution simulates clusters that arise from skewed distributions, and the multivariate normal distribution with unequal variances results in elliptical clusters. For each of the data sets considered in the current experiment, the underlying distribution of the cluster structure was randomly chosen (with equal probability) from one of these five scenarios.

Once the distribution of the clusters was determined, the average amount the clusters overlap was allowed to vary from $O = 0$, .20, and .40 and was implemented using the OCLUS generation procedure described by Steinley and Henson (2005). Another slight complexity that must be considered is the type of overlap imposed upon the cluster. Steinley and Henson (2005) describe two types of overlap, marginal and joint. Marginal overlap is defined as overlap that occurs independently on each variable that describes the cluster structure, whereas joint overlap is defined as the amount of overlap that occurs in the joint variable space. In other words, under marginal overlap, clusters are only guaranteed to overlap on each variable; however, there still might be spaces of low density between the clusters in the multivariate space. Under joint overlap, the clusters are guaranteed to overlap on all variables. Logically, it should be more difficult for clustering algorithms to recover cluster structure when overlap is guaranteed to simultaneously exist on all dimensions. In fact, this has been shown to be the case regardless of the underlying distributional family the clusters are generated from (see Steinley, 2006b). For the present experiment, the type of overlap for each data set was randomly chosen to be either marginal or joint.

3.2. Implementation

The experimental study was conducted using MatLab Version 6.5 (MathWorks, Inc., 2002) on a 2.6 GHz Pentium IV PC with 1 GB of RAM. All heuristic methods were written as MatLab

*.m files, as were batch files for problem generation and data collection. Manipulation of the experimental test conditions resulted in a total of $3 \times 4 \times 3 \times 3 = 108$ cells (number of clusters $\times$ number of variables $\times$ relative density $\times$ overlap). Like previous Monte Carlo experiments (Brusco, 2004; Brusco & Cradit, 2001; Milligan, 1980a; Steinley, 2006b), three data sets were generated per cell, resulting in 324 unique data sets to be analyzed. After generation of each test problem, each heuristic method was applied to the problem subject to a limitation of 3 CPU minutes, resulting in 27 minutes of CPU time for each data set and a total of 8,748 minutes (145.8 hours or just over 6 days) of computation for the entire experiment. The same set of initial partitions was used for multiple restarts of each heuristic, and a new restart of a heuristic was not allowed to begin if the 3-minute limit was exceeded. The HK-means algorithm was used to post-process the simulated annealing, tabu search, and genetic algorithm solutions, so as to ensure these methods produced local minima.

The primary piece of information collected for each heuristic and each test problem was the partition producing the best WCSS value during the prescribed duration. We denote π_K^e as the partition with the best WCSS value, and $WCSS(\pi_K^e)$ as the corresponding best WCSS value obtained by method e ($1 \leq e \leq 9$) within the time limit. The best WCSS value across all methods is obtained as $WCSS(\pi_K^{\text{Best}}) = \min_{1 \leq e \leq 9}(WCSS(\pi_K^e))$ and π_K^{Best} is the corresponding partition with the best WCSS value across all methods. This enables the WCSS value obtained by each method to be expressed as a percentage deviation above the best WCSS value found across all methods as follows:

$$WCSS_{\text{dev}}(e) = 100 \times \frac{(WCSS(\pi_K^e) - WCSS(\pi_K^{\text{Best}}))}{WCSS(\pi_K^{\text{Best}})} \quad \text{for } 1 \leq e \leq 9. \tag{5}$$

Although $WCSS_{\text{dev}}(e)$ is the primary performance measure in our experiments, we also consider a secondary performance measure related to cluster recovery. In particular, for each method and each test problem, we compute Hubert and Arabie's (1985) adjusted Rand index (ARI) between the partition associated with the best WCSS value produced by method e, π_K^e, and the true cluster structure associated with the generation of the test problem. These values are denoted $ARI(e)$ for $1 \leq e \leq 9$. A value of $ARI(e) = 1$ indicates perfect agreement between π_K^e and the partition corresponding to the true cluster structure, whereas a value of $ARI(e) = 0$ indicates chance agreement between the two partitions. Steinley (2004) suggested that values of .65 and .80 indicate adequate and good agreement between two partitions, respectively.

3.3. Overall Performance

Table 1 provides the overall performance of each of the methods in terms of $WCSS_{\text{dev}}(e)$. The genetic algorithm procedure, when used in conjunction with HK-means as a post-processing procedure, performed best across all conditions. The variable neighborhood search method, J-means+, and HK-means were the only remaining procedures to return WCSS values within 1% of the genetic algorithm solution across all the factor levels. Furthermore, KI-means, tabu search, and simulated annealing only performed slightly worse, returning WCSS values between 1%–3% greater on average than those obtained by the genetic algorithm. Finally, it appears that H-means+ and K-means performed abysmally compared to the genetic algorithm, with solutions that are 25% and 29% worse on average, respectively.

Table 1 also provides some indication of how the relative performances of methods are affected by different test problem conditions. For example, the performances of most algorithms generally worsened as K increased. Four algorithms, H-means+, HK-means, J-means+, and variable neighborhood search always matched $WCSS(\pi_K^{\text{Best}})$ when $K = 4$; however, their relative performances worsened slightly when moving to $K = 6$ and, more significantly, at $K = 8$. The

TABLE 1.
Percentage deviation from best-found WCSS value.

Condition*	Method								
	H-means+	K-means	HK-means	KI-means	J-means+	Tabu	SA	Genetic	VNS
$K = 4$	.00	9.06	.00	.73	.00	.43	.36	.00	.00
$K = 6$	13.75	32.70	.17	2.72	.10	1.37	1.85	.00	.15
$K = 8$	60.18	45.82	.83	4.22	.66	1.29	.82	.00	.58
$V = 4$	43.88	30.42	.21	2.27	.49	.26	.17	.00	.16
$V = 6$	30.31	26.07	.38	2.34	.18	1.07	.88	.00	.30
$V = 8$	.15	14.26	.24	1.07	.12	.58	.43	.00	.20
$V = 10$	24.22	46.03	.50	4.54	.23	2.21	2.56	.00	.32
$D = 1$	25.20	36.28	.41	2.24	.33	.06	.01	.00	.41
$D = 2$	32.38	28.07	.34	1.52	.34	.04	.03	.00	.30
$D = 3$	16.35	23.24	.25	3.90	.09	3.00	2.99	.00	.03
$O = .00$	62.12	80.93	.99	7.38	.67	2.99	2.98	.00	.72
$O = .20$	2.60	4.46	.01	.20	.06	.07	.03	.00	.01
$O = .40$	9.20	2.19	.00	.08	.04	.04	.02	.00	.00
Average	24.64	29.19	.33	2.55	.25	1.03	1.01	.00	.24

*For each row, the results are averaged across the other conditions.

density of clusters seemed to have a strong effect on the performances of some clustering methods, but not others. For example, the tabu search and simulated annealing heuristics performed well at settings of $D = 1$ and $D = 2$. These heuristics were less effective for the setting of $D = 3$, indicating that their performances deteriorated when there was one large cluster and several smaller clusters.

The degree of cluster overlap also had a pronounced effect on WCSS optimization performance. The largest percentage deviations from the best WCSS values occurred when there was no cluster overlap ($O = .00$). Although this finding might initially seem counterintuitive, it is readily explainable by considering the effect of misclassification of objects. In the cases of greater cluster overlap ($O = .20$ and $O = .40$), misclassification of even a large number of objects does not necessarily have a significant impact on WCSS because the cluster centroids are not as separated. In contrast, when just a few objects are misclassified in a well-separated data set with no cluster overlap ($O = .00$), the effect on WCSS can be profound because of the distortion of the cluster centroids.

Table 2 provides the average ARI of each method across the various conditions. As well as performing the best in minimizing WCSS, the genetic algorithm also had the highest degree of true cluster recovery. This finding provides some evidence that the WCSS criterion is effective for recovering cluster structure across a broad range of data conditions. Moreover, as was the case for WCSS, HK-means and the variable neighborhood search procedure were close behind the genetic algorithm with respect to cluster recovery, yielding an overall ARI average within .005 of corresponding average for the genetic algorithm. More surprisingly, although H-means+ finished eighth in average $WCSS_{\text{dev}}(e)$ performance (approximately 24% worse in terms of minimizing the objective function), the average ARI value for H-means+ was only .011 less than that observed for the genetic algorithm. The average ARI for J-means+ was only slightly less than those of the best methods, and, as with WCSS, the remaining four procedures (K-means, KI-means, tabu search, and simulated annealing) performed the worst of the nine methods.

With respect to the effect of experimental test conditions on the relative ARI performances of the algorithms, perhaps the most critical factor was the degree of cluster overlap. Note the contrast between the rows for $O = .00$, $O = .20$, and $O = .40$ in Table 1 with the corresponding

TABLE 2.
Partition agreement—average adjusted Rand index with true cluster solution.

	Method								
Condition*	H-means+	K-means	HK-means	KI-means	J-means+	Tabu	SA	Genetic	VNS
$K = 4$	.719	.528	.719	.630	.718	.657	.676	.719	.719
$K = 6$	.707	.446	.714	.591	.689	.652	.658	.719	.712
$K = 8$	.609	.389	.645	.485	.609	.589	.603	.667	.650
$V = 4$	.655	.437	.658	.573	.647	.610	.627	.658	.658
$V = 6$	.727	.506	.735	.632	.705	.669	.700	.734	.727
$V = 8$	.755	.516	.750	.641	.740	.694	.702	.760	.755
$V = 10$	.679	.456	.696	.559	.683	.627	.627	.703	.698
$D = 1$	.761	.395	.776	.707	.766	.792	.787	.783	.777
$D = 2$	.698	.400	.693	.654	.682	.712	.708	.696	.695
$D = 3$	.645	.634	.657	.429	.631	.439	.485	.659	.652
$O = .00$	.883	.567	.899	.734	.878	.812	.817	.909	.896
$O = .20$	.639	.422	.639	.545	.629	.591	.607	.641	.640
$O = .40$	.581	.440	.588	.510	.572	.539	.556	.589	.588
Average	.702	.476	.709	.596	.692	.647	.660	.713	.708

*For each row, the results are averaged across the other conditions.

rows in Table 2. Although the average WCSS percentage deviations were greatest for $O = .00$, Table 2 shows that the recovery was best at this setting. Again, this finding is attributable to the fact that a few misplaced objects can tremendously inflate WCSS percentage error, yet have only a modest effect on recovery. For the conditions of greater overlap, WCSS error is small, but a greater number of misclassified points diminished recovery.

A close inspection of relative recovery performance at the setting for no overlap ($O = .00$) is especially important because this is a data scenario where recovery should be strong. The genetic algorithm had an average ARI of .909 at this setting, and was followed closely by HK-means (.899), variable neighborhood search (.896), H-means+ (.883), and J-means+ (.878). Each of the other algorithms produced an ARI of .817 or smaller, which suggests that meaningful differences in recovery can occur even when clusters are well separated.

4. Computational Experiments for Empirical Data Sets

4.1. The Data Sets

We also investigated the performances of the WCSS partitioning algorithms for 13 data sets collected from the literature. Our goal was to select a set of test problems that exhibits variation with respect to the number of objects, the number of variables, and the context of the data set. Eight of the 13 data sets are two-dimensional, and correspond to contexts such as city coordinates, birth/death rates, and hole-drilling positions for printed circuit boards. The two-dimensional problems range in size from $N = 59$ to $N = 1060$ objects. The number of dimensions for the five remaining data sets ranges from $3 \leq V \leq 7$. Measurements for these data sets include plant characteristics, body dimensions, vowel sounds, and Likert-scale values for various marketing profiles. The selected data sets are as follows:

- Data Set I. Two-dimensional coordinates for $N = 59$ German towns as recorded by Späth (1980, p. 80).

- Data Set II. Birth and death rates ($V = 2$) for $N = 70$ countries as reported by Hartigan (1975, Chap. 11).
- Data Set III. Area, population, and population density ($V = 3$) for $N = 89$ Bavarian postal codes from Späth (1980, pp. 91–92).
- Data Set IV. The HATCO data. A synthetic data set consisting of $V = 7$ measurements for $N = 100$ objects, as published by Hair, Anderson, Tatham, and Black (1998, pp. 725–726).
- Data Set V. Iris data. The data consist of $V = 4$ measurements for each of $N = 150$ plants, and were originally collected by Anderson (1935) and subsequently analyzed by Fisher (1936).
- Data Set VI. Grötschel and Holland's (1991) European city coordinates data, part A. Two-dimensional coordinates for $N = 202$ cities. These data were obtained from a library of traveling salesman problems, TSPLIB (Reinelt, 2001).
- Data Set VII. Grötschel and Holland's (1991) European city coordinates data, part B. Two-dimensional coordinates for $N = 431$ cities. These data were obtained from a library of traveling salesman problems, TSPLIB (Reinelt, 2001).
- Data Set VIII. Grötschel and Holland's (1991) European city coordinates data, part C. Two-dimensional coordinates for $N = 666$ cities. These data were obtained from a library of traveling salesman problems, TSPLIB (Reinelt, 2001).
- Data Set IX. Reinelt's (2001) data corresponding to hole-drilling for printed circuit boards. The data correspond to $N = 724$ hole positions in two-dimensional space and were obtained from TSBLIB (Reinelt, 2001).
- Data Set X. Reinelt's (2001) data corresponding to hole-drilling for printed circuit boards. The data correspond to $N = 1060$ hole positions in two-dimensional space and were obtained from TSBLIB (Reinelt, 2001).
- Data Set XI. Body measurements data. These data were extracted from a body measurements data set collected by Heinz, Peterson, Johnson, and Kerk (2003). The data, which correspond to $V = 5$ body measurements (weight, height, chest girth, waist girth, and hips girth) measured for $N = 507$ men and women, were obtained electronically from: [www.amstat.org/publications/jse/v11n2/datasets.heinz.html].
- Data Set XII. Indian Telugu vowel sounds. These data consist of $N = 871$ measurements for $V = 3$ vowel formant frequencies, and have been studied by Pal and Majumder (1977) and Maulik and Bandyopadhyay (2000).
- Data Set XIII. Telecommunications data. These data were collected by Brusco, Cradit, and Tashchian (2003) and correspond to responses from $N = 475$ respondents with respect to $V = 5$ performance features related to their current long-distance provider: (a) competitively priced products; (b) responsive account executives; (c) billing accuracy; (d) state-of-the-art products; and (e) responsive repair service.

4.2. Experimental Results for Empirical Data Sets

We applied each of the nine WCSS partitioning algorithms to each of the 13 empirical data sets under the assumption of $K = 5$ clusters and $K = 10$ clusters. Our choice of the levels for K was based on a desire to select plausible values for social sciences applications, as well as to differentiate the heuristics based on WCSS performance. An important caveat, however, is that these levels do not necessarily represent the best values of K for the 13 data sets. For consistency with the experiments on simulated data sets reported in the previous section, each algorithm was limited to 3 minutes of microcomputer CPU time and the best WCSS value obtained by each algorithm was collected and stored along with the corresponding partition. The HK-means algorithm was used to post-process the simulated annealing, tabu search, and genetic algorithm solutions, so as to ensure these methods produced local minima.

TABLE 3.
Percentage deviation from best-found WCSS value ($K = 5$).

	Method								
Data set	H-means+	K-means	HK-means	KI-means	J-means+	Tabu	SA	Genetic	VNS
I	.00	.00	.00	.00	.00	.00	.00	.00	.00
II	.00	.00	.00	.00	.00	.00	.00	.00	.00
III	44.66	.00	.00	.00	.00	44.66	44.66	44.66	.00
IV	.00	.00	.00	.00	.00	.00	.00	.00	.00
V	.00	.00	.00	.00	.00	.00	.00	.00	.00
VI	.00	.00	.00	.00	.00	.00	.00	.00	.00
VII	.00	.00	.00	.54	.00	.00	.00	.00	.00
VIII	.00	3.83	.00	3.83	.00	.00	3.83	.00	.00
IX	.00	.00	.00	.00	.00	.00	.00	.00	.00
X	.00	.00	.00	.00	.01	.00	.00	.00	.00
XI	.01	.01	.01	.01	.01	.01	.00	.00	.00
XII	.01	.00	.00	.00	.09	.00	.00	.00	.00
XIII	.01	.00	.00	.05	.00	.04	.00	.00	.00
Average	3.44	.30	.00	.34	.01	3.44	3.73	3.44	.00
Max	44.66	3.83	.01	3.83	.09	44.66	44.66	44.66	.00

TABLE 4.
Partition agreement—adjusted Rand index with best found partition ($K = 5$).

	Method								
Data set	H-means+	K-means	HK-means	KI-means	J-means+	Tabu	SA	Genetic	VNS
I	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
II	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
III	.841	1.000	1.000	1.000	1.000	.841	.841	.841	1.000
IV	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
V	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
VI	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
VII	1.000	1.000	1.000	.976	1.000	1.000	1.000	1.000	1.000
VIII	1.000	.756	1.000	.756	1.000	1.000	.756	1.000	1.000
IX	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
X	1.000	1.000	1.000	.990	.971	1.000	1.000	1.000	1.000
XI	.994	.856	.994	.856	.856	.856	1.000	1.000	1.000
XII	.992	1.000	1.000	.992	.952	1.000	1.000	1.000	1.000
XIII	.995	1.000	1.000	.917	1.000	.933	1.000	1.000	1.000
Average	.986	.970	.999	.961	.983	.972	.969	.988	1.000
Min	.841	.756	.994	.756	.856	.841	.756	.841	1.000

The results for $K = 5$ are presented in Tables 3 and 4. Table 3 reports the $WCSS_{\text{dev}}(e)$ values for each method on each test problem. The results indicate that all methods performed well, with each method matching $WCSS(\pi_K^{\text{Best}})$ for at least 9 of the 13 test problems. The variable neighborhood search heuristic was the only heuristic that matched $WCSS(\pi_K^{\text{Best}})$ for all 13 problems; however, its performance was followed closely by HK-means and the genetic algorithm, which each produced 12 best WCSS values. The results for data set III are especially interesting. Five of the nine methods identified the best-found partition, which we know is optimal from the work of du Merle et al. (2000). The four methods that failed to find the optimal partition, yielded a local minimum with a departure of more than 44%. As observed by Hansen and Mladenoviĉ (2001),

TABLE 5.
Percentage deviation from best-found WCSS value ($K = 10$).

	Method								
Data set	H-means+	K-means	HK-means	KI-means	J-means+	Tabu	SA	Genetic	VNS
I	5.92	.00	1.66	.00	1.21	.00	.00	.00	.00
II	.53	3.77	.04	3.77	.04	.00	.00	.00	.00
III	419.17	185.94	.00	185.94	12.29	.00	.00	.00	.00
IV	1.97	.00	.00	.00	.00	.00	.00	.00	.00
V	16.70	24.98	15.82	24.98	1.49	.00	.00	.00	.00
VI	.52	6.40	.00	6.40	.00	3.80	3.80	.00	.00
VII	2.58	2.44	2.02	2.44	6.75	4.04	2.80	.00	.00
VIII	10.02	5.77	5.79	14.20	.00	9.71	9.59	.00	.00
IX	.01	.00	.00	.01	.00	1.11	.84	.00	.00
X	.03	.19	.19	.19	.22	2.71	.19	.00	.00
XI	.25	.22	.22	.22	.28	.22	.16	.00	.00
XII	2.50	.16	2.35	4.01	.98	3.81	.00	.00	.00
XIII	.51	.48	.05	1.75	2.29	.73	.15	.07	.00
Average	35.44	17.72	2.16	18.76	1.97	2.01	1.35	.01	.00
Max	419.17	185.94	15.82	185.94	12.29	9.71	9.59	.07	.00

the neighborhoods of optimal partitions for this data set are often rugged and well-protected by poor local minima.

Table 4 reports $ARI(e)$ values for each heuristic and each test problem under the assumption of $K = 5$ clusters. Unlike the simulated experiments reported in Section 3, there is no "true" cluster structure for the empirical data sets. Therefore, the $ARI(e)$ values in Table 4 represent, for each data set, the agreement between π_K^{Best} and π_K^e for $(1 \leq e \leq 9)$. All methods have an average (across the 13 data sets) ARI value of at least .969, and the minimum ARI value across all problems and methods was .756. Even the heuristics that exhibited serious departure from the optimal WCSS value for data set III had very reasonable agreement with the optimal partition, as evidenced by the ARI value of .841.

Table 5 reports the $WCSS_{\text{dev}}(e)$ values for each method on each of the 13 data sets under the assumption $K = 10$ clusters. The results in Table 5 indicate that the increase in the number of clusters from 5 to 10 resulted in a serious deterioration in relative performance for many of the heuristic methods. For example, H-means+ does not match $WCSS(\pi_K^{\text{Best}})$ for any of the 13 data sets, and K-means, HK-means and KI-means, each produced a partition with the best WCSS value for three or fewer problems.

The variable neighborhood search heuristic and genetic algorithm were the best performing methods across the test problems. The variable neighborhood search matched $WCSS(\pi_K^{\text{Best}})$ for each of the 13 test problems and the genetic algorithm matched the best found solution for all but one test problem, exhibiting a .07% departure from $WCSS(\pi_K^{\text{Best}})$ for data set XIII. The simulated annealing heuristic was a distant third in performance, matching $WCSS(\pi_K^{\text{Best}})$ for only 6 of the 13 data sets, and producing a departure of 2.5% or greater for three of the data sets.

The resulting $ARI(e)$ values for each test problem under the assumption of $K = 10$ clusters are displayed in Table 6. Like Table 4, the ARIs for variable neighborhood search are all 1 because this algorithm always produced the partition with the best WCSS value. Because it matched the variable neighborhood search solution for all but one problem, the genetic algorithm ARIs are all 1 except for test problem XIII; however, the ARI of .983 for this problem suggests that the genetic algorithm partition was not appreciably different than the best-found partition. The genetic algorithm and variable neighborhood search performances notwithstanding, the best WCSS partitions produced by the remaining algorithms occasionally differed markedly from the

TABLE 6.
Partition agreement—adjusted Rand index with best-found partition ($K = 10$).

	Method								
Data set	H-means+	K-means	HK-means	KI-means	J-means+	Tabu	SA	Genetic	VNS
I	.681	1.000	.816	1.000	.854	1.000	1.000	1.000	1.000
II	.796	.739	.782	.739	.782	1.000	1.000	1.000	1.000
III	.802	.949	1.000	.949	.795	1.000	1.000	1.000	1.000
IV	.761	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
V	.678	.616	.738	.616	.850	1.000	1.000	1.000	1.000
VI	.919	.626	1.000	.626	.970	.828	.828	1.000	1.000
VII	.857	.911	.831	.911	.686	.582	.871	1.000	1.000
VIII	.720	.904	.918	.806	1.000	.675	.639	1.000	1.000
IX	.984	1.000	1.000	.984	1.000	.739	.804	1.000	1.000
X	.917	.732	.732	.732	.878	.693	.732	1.000	1.000
XI	.696	.691	.691	.691	.671	.691	.800	1.000	1.000
XII	.688	.923	.699	.773	.778	.762	1.000	1.000	1.000
XIII	.564	.665	.947	.627	.662	.647	.793	.983	1.000
Average	.774	.827	.858	.804	.840	.817	.882	.999	1.000
Min	.564	.616	.691	.616	.662	.582	.639	.983	1.000

best-found WCSS partitions across all methods. Each of these algorithms produced an ARI value of .7 or less for one or more test problems, and there were numerous ARIs below .8.

5. Discussion

5.1. Summary of Main Findings

Among the heuristic methods popularized in the 1960s and 1970s for MWCSSP, our results indicate that HK-means is superior with respect to WCSS performance, as well as recovery of cluster structure. HK-means outperformed H-means+, K-means, and KI-means in the experiments for both simulated and real data sets. Our multistart implementation of HK-means was also very competitive with more sophisticated procedures, often outperforming both the tabu search and simulated annealing implementations. It is our opinion that researchers in the behavioral sciences should have considerable confidence in a multistart implementation of HK-means when $K \leq 6$. In our experiment using simulated data sets, HK-means produced an average ARI of .717 when $K \leq 6$, which was second only to the genetic algorithm. For the experiment using real data sets, HK-means obtained the best-found WCSS value for 12 of the 13 test problems at $K = 5$ with only a small percentage error for data set XI.

For $K > 6$, our experiments indicate that Maulik and Bandyopadhyay's (2000) genetic algorithm and Hansen and Mladenović's (2001) variable neighborhood search procedure generally outperform multistart HK-means; however, it should be recalled that both of these latter methods used the HK-means algorithm in our implementation. The genetic algorithm was the best-performing procedure in the experiment using simulated data with respect to both minimization of WCSS and recovery of true cluster structure. For the empirical data sets, the genetic algorithm matched the best WCSS value obtained by the variable neighborhood search algorithms for 12 of the 13 test problems at $K = 5$, and 12 of 13 problems when $K = 10$. We cannot declare a clear winner between the genetic algorithm and the variable neighborhood search method; however, the two procedures were clearly the best performers when $K > 6$ and we confidently recommend them in such circumstances.

5.2. *Caveats and Limitations*

We attempted to select a broad representation of solution procedures for our comparative evaluations. Although a concerted effort was made to select from among the best procedures available in the literature, there were many difficult choices and some caution is advisable when interpreting the results. It is important to remember that the results of our analyses are based on specific implementations of each method, and should not be extended to compare the broader methodological choices. For example, our results showed that a particular implementation of a genetic algorithm outperformed a particular implementation of simulated annealing. This does not imply that all implementations of genetic algorithms are better than all implementations of simulated annealing. The disparity in the results is more likely attributable to the neighborhood jump moves available in the genetic algorithm, which could easily be included in a simulated annealing algorithm. We cannot guarantee that our implementation of any method is the best possible, there are too many options.

We conducted our study in a MatLab environment and all programs were written by the authors, which mitigates the potential for considerable disparity in the efficiency of the codes. Nevertheless, our results would not necessarily generalize to different computer hardware and software platforms. For example, if other researchers replicated this study with the same CPU time limits, but using a Fortran implementation on a faster machine, the results could be different. The number of solutions produced by some of the more efficient procedures could increase by several orders of magnitude, thus improving their relative performances.

We also acknowledge that there are a variety of follow-up experiments that would nicely augment the results of our experiments. For example, in our experiments we randomly produced the initial partitions for each restart of the algorithms. However, there are numerous other available approaches for creating initial partitions (Hand & Krzanowski, 2005; Milligan, 1980b; Steinley, 2003) and an evaluation of these approaches in conjunction with one or more of the clustering algorithms studied herein is a worthy avenue of investigation (see Steinley & Brusco, 2007). Also, as noted by Steinley (2006a), it is well recognized that the WCSS criterion is appropriate for spherical clusters of approximately the same size. For data sets where clusters have more of an elliptical shape, possibly with different spatial orientations and cluster sizes, other criteria might be preferred (Banfield & Raftery, 1993; Maronna & Jacovkis, 1974; Scott & Symons, 1971). Under such conditions, a comparison of cluster recovery that focuses on comparison of different partitioning *criteria* rather than different partitioning *algorithms* is, therefore, also highly worthwhile.

References

Al-Sultan, K. (1995). A tabu search approach to the clustering problem. *Pattern Recognition*, *28*, 1443–1451.

Anderson, E. (1935). The irises of the Gaspé peninsula. *Bulletin of the American Iris Society*, *59*, 2–5.

Arabie, P., & Hubert, L. (1992). Combinatorial data analysis. *Annual Review of Psychology*, *43*, 169–203.

Arabie, P., & Hubert, L.J. (1996). An overview of combinatorial data analysis. In P. Arabie, L.J. Hubert, & G. De Soete (Eds.), *Clustering and classification* (pp. 5–63). River Edge: World Scientific.

Babu, G.P., & Murty, M.N. (1993). A near optimal initial seed value selection in k-means algorithm using genetic algorithms. *Pattern Recognition Letters*, *14*, 763–769.

Babu, G.P., & Murty, M.N. (1994). Simulated annealing for selecting optimal initial seeds in the K-means algorithm. *Indian Journal of Pure and Applied Mathematics*, *25*, 85–94.

Banfield, C.F., & Bassil, L.C. (1977). A transfer algorithm for nonhierarchical classification. *Applied Statistics*, *26*, 206–210.

Banfield, J.D., & Raftery, A.E. (1993). Model-based Gaussian and non-Gaussian clustering. *Biometrics*, *49*, 803–821.

Belew, R.K., & Booker, J.B. (Eds.). (1991). *Proceedings of the fourth international conference on genetic algorithms*. San Mateo: Morgan-Kaufmann.

Brucker, F. (1978). On the complexity of clustering problems. In M. Beckmann & H.P. Kunzi (Eds.), *Optimization and operations research* (pp. 45–54). Heidelberg: Springer.

Brusco, M.J. (2004). Clustering binary data in the presence of masking variables. *Psychological Methods*, *9*, 510–523.

Brusco, M.J. (2006). A repetitive branch-and-bound procedure for minimum within-cluster sums of squares partitioning. *Psychometrika*, *71*, 347–363.

Brusco, M.J., & Cradit, J.D. (2001). A variable selection heuristic for *k*-means cluster analysis. *Psychometrika*, *66*, 249–270.

Brusco, M.J., Cradit, J.D., & Tashchian, A. (2003). Multicriterion clusterwise regression for joint segmentation settings: An application to customer value. *Journal of Marketing Research*, *40*, 225–234.

Cerny, V. (1985). A thermodynamical approach to the traveling salesman problem. *Journal of Optimization Theory and Applications*, *45*, 41–51.

Day, W.H.E. (1996). Complexity theory: An introduction for practitioners of classification. In P. Arabie, L.J. Hubert, & G. De Soete (Eds.), *Clustering and classification* (pp. 199–233). River Edge: World Scientific.

Diehr, G. (1985). Evaluation of a branch and bound algorithm for clustering. *SIAM Journal for Scientific and Statistical Computing*, *6*, 268–284.

Dimitriadou, E., Dolnicâr, S., & Weingessel, A. (2002). An examination of indexes for determining the number of clusters in binary data sets. *Psychometrika*, *67*, 137–160.

du Merle, O., Hansen, P., Jaumard, B., & Mladenovicâ, N. (2000). An interior point algorithm for minimum sum-of-squares clustering. *SIAM Journal on Scientific Computing*, *21*, 1485–1505.

Fisher, R.A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, *7*, 179–188.

Forgy, E.W. (1965). Cluster analyses of multivariate data: Efficiency versus interpretability of classifications. *Biometrics*, *21*, 768.

Forrest, S. (Ed.). (1993). *Proceedings of the fifth international conference on genetic algorithms*. San Mateo: Morgan-Kaufmann.

Glover, F. (1989). Tabu search—Part I. *ORSA Journal on Computing*, *1*, 190–206.

Glover, F. (1990). Tabu search—Part II. *ORSA Journal on Computing*, *2*, 4–32.

Glover, F., & Laguna, M. (1993). Tabu search. In C. Reeves (Ed.), *Modern heuristic techniques for combinatorial problems* (pp. 70–141). Oxford: Blackwell.

Glover, F., Taillard, E., & Werra, D. (1993). A user's guide to tabu search. *Annals of Operations Research*, *41*, 3–28.

Goldberg, D.E. (1989). *Genetic algorithms in search, optimization, and machine learning*. New York: Addison-Wesley.

Grötschel, M., & Holland, O. (1991). Solution of large-scale symmetric traveling salesman problems. *Mathematical Programming*, *51*, 141–202.

Hair, J.F., Anderson, R.E., Tatham, R.L., & Black, W.C. (1998). *Multivariate data analysis* (5th ed.). Saddle River: Prentice-Hall.

Hand, D.J. (1981). *Discrimination and classification*. New York: Wiley.

Hand, D.J., & Krzanowski, W.J. (2005). Optimising *k*-means clustering results with standard software packages. *Computational Statistics and Data Analysis*, *49*, 969–973.

Hansen, P., & Mladenovicâ, N. (2001). *J*-Means: A new local search heuristic for minimum sum of squares clustering. *Pattern Recognition*, *34*, 405–413.

Hartigan, J.A. (1975). *Clustering algorithms*. New York: Wiley.

Hartigan, J.A., & Wong, M.A. (1979). Algorithm AS136: A *k*-means clustering program. *Applied Statistics*, *28*, 100–128.

Heinz, G., Peterson, L.J., Johnson, R.W., & Kerk, C.J. (2003). Exploring relationships in body dimensions. *Journal of Statistics Education*, *11*, www.amstat.org/publications/jse/v11n2/datasets.heinz.html.

Holland, J.H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press.

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification*, *2*, 193–218.

Hubert, L., Arabie, P., & Meulman, J. (2001). *Combinatorial data analysis: Optimization by dynamic programming*. Philadelphia: Society Industrial and Applied Mathematics.

Jancey, R.C. (1966). Multidimensional group analysis. *Australian Journal of Botany*, *14*, 127–130.

Jones, D.R., & Beltramo, M.A. (1991). Solving partitioning problems with genetic algorithms. In R.K. Belew & J.B. Booker (Eds.), *Proceedings of the fourth international conference on genetic algorithms*. San Mateo: Morgan Kaufmann.

Kirkpatrick, S., Gelatt, C.D., & Vecchi, M.P. (1983). Optimization by simulated annealing. *Science*, *220*, 671–680.

Klein, R.W., & Dubes, R.C. (1989). Experiments in projection and clustering by simulated annealing. *Pattern Recognition*, *22*, 213–220.

Koontz, W.L.G., Narendra, P.M., & Fukunaga, K. (1975). A branch and bound clustering algorithm. *IEEE Transaction on Computers*, *C-24*, 908–915.

Krishna, K., & Murty, N.M. (1999). Genetic *K*-means algorithm. *IEEE Transactions on Systems, Man, & Cybernetics—Part B: Cybernetics*, *29*, 433–439.

MacQueen, J.B. (1967). Some methods for classification and analysis of multivariate observations. In L.M. Le Cam & J. Neyman (Eds.), *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (Vol. 1, pp. 281–297). Berkeley: University of California Press.

Maronna, R., & Jacovkis, P.M. (1974). Multivariate clustering procedures with variable metrics. *Biometrics*, *30*, 499–505.

MathWorks, Inc. (2002). *Using MATLAB (version 6)*. Natick: The MathWorks, Inc.

Maulik, U., & Bandyopadhyay, S. (2000). Genetic algorithm-based clustering technique. *Pattern Recognition*, *33*, 1455–1465.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A., & Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, *21*, 1087–1092.

Milligan, G.W. (1980a). An examination of the effect of six types of error perturbation on fifteen clustering algorithms. *Psychometrika*, *45*, 325–342.

Milligan, G.W. (1980b). The validation of four ultrametric clustering algorithms. *Pattern Recognition*, *12*, 41–50.
Milligan, G.W. (1985). An algorithm for generating artificial test clusters. *Psychometrika*, *50*, 123–127.
Milligan, G.W. (1989). A validation study of a variable-weighting algorithm for cluster analysis. *Journal of Classification*, *6*, 53–71.
Milligan, G.W., & Cooper, M.C. (1986). A study of the comparability of external criteria for hierarchical cluster analysis. *Multivariate Behavioral Research*, *21*, 441–458.
Milligan, G.W., & Cooper, M.C. (1988). A study of variable standardization. *Journal of Classification*, *5*, 181–204.
Pacheco, J., & Valencia, O. (2003). Design of hybrids for the minimum sum-of-squares clustering problem. *Computational Statistics and Data Analysis*, *43*, 235–248.
Pal, S.K., & Majumder, D.D. (1977). Fuzzy sets and decision making approaches in vowel and speaker recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, *7*, 625–629.
Reinelt, G. (2001). TSPLIB. http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95.
Scott, A.J., & Symons, M.J. (1971). Clustering methods based on likelihood ratio criteria. *Biometrics*, *27*, 387–398.
Selim, S.Z., & Al-Sultan, K. (1991). A simulated annealing algorithm for the clustering problem. *Pattern Recognition*, *24*, 1003–1008.
Späth, H. (1980). *Cluster analysis algorithms for data reduction and classification of objects*. New York: Wiley.
Steinley, D. (2003). Local optima in K-means clustering: What you don't know may hurt you. *Psychological Methods*, *8*, 294–304.
Steinley, D. (2004). Properties of the Hubert–Arabie adjusted Rand index. *Psychological Methods*, *9*, 386–396.
Steinley, D. (2006a). K-means clustering: A half-century synthesis. *British Journal of Mathematical and Statistical Psychology*, *59*, 1–34.
Steinley, D. (2006b). Profiling local optima in K-means clustering: Developing a diagnostic technique. *Psychological Methods*, *11*, 178–192.
Steinley, D., & Brusco, M. (2007). Initializing K-means batch clustering: A critical analysis of several techniques. *Journal of Classification*, *24*, 99–121.
Steinley, D., & Henson, R. (2005). OCLUS: An algorithmic method for generating clusters with known overlap. *Journal of Classification*, *22*, 221–250.
Sun, L.-X., Xie, Y.-L., Song, X.-H., Wang, J.-H., & Yu, R.-Q. (1994a). Cluster analysis by simulated annealing. *Computers in Chemistry*, *18*, 103–108.
Sun, L.-X., Xu, F., Liang, Y.-Z., Xie, Y.-L., & Yu, R.-Q. (1994b). Cluster analysis by the K-means algorithm and simulated annealing. *Chemometrics and Intelligent Laboratory Systems*, *25*, 51–60.
Sung, C.S., & Jin, H.W. (2000). A tabu-search-based heuristic for clustering. *Pattern Recognition*, *33*, 849–858.
van Os, B.J., & Meulman, J.J. (2004). Improving dynamic programming strategies for partitioning. *Journal of Classification*, *21*, 207–230.

Manuscript received 20 JUL 2005
Final version received 8 DEC 2006