A Practical Application of Simulated Annealing to Clustering

Donald E. Brown
Christopher L. Huntley
IPC-TR-91-003

2/11/91

Institute for Parallel Computation
and
Department of Systems Engineering
University of Virginia
Charlotesville, VA 22901

Email: deb@virginia.edu and clh1n@virginia.edu

Acknowledgement

This work was supported in part by the Jet Propulsion Laboratory under grant number 95722, and CSX Transportation under grant number 5-38426

Abstract

We formalize clustering as a partitioning problem with a user-defined internal clustering criterion and present SINICC, an unbiased, empirical method for comparing internal clustering criteria. An application to multi-sensor fusion is described, where the data set is composed of inexact sensor "reports" pertaining to "objects" in an environment. Given these reports, the objective is to produce a representation of the environment, where each entity in the representation is the result of "fusing" sensor reports. Before one can perform fusion, however, the reports must be "associated" into homogeneous clusters. Simulated annealing is used to find a near-optimal partitioning with respect to each of several clustering criteria for a variety of simulated data sets. This method can then be used to determine the "best" clustering criterion for the multi-sensor fusion problem with a given fusion operator.

1. Introduction

Clustering is a process basic to human understanding. The grouping of related objects can be found in such diverse fields as statistics, economics, physics, psychology, biology, pattern recognition, engineering, and marketing. Since its range of application is so large, there is no "fundamental" clustering problem formulation because the relationships between the objects can vary. We use simulated annealing to "solve" a very general formulation of the problem. Since simulated annealing works well on a wide range of combinatorial problems, it would seem that clustering is a natural application. However, in a previous study (Klein and Dubes 1989), simulated annealing provided good clusterings, but proved impractical for repeated use on large clustering problems because of the computational effort involved. We present a practical application of simulated annealing to clustering.

Two domain-specific details are common to most clustering problem formulations: 1) a *data structure* used to define clusters and 2) an *internal clustering criterion* based on a model of the clusters expected in the domain. Whereas previous studies have applied simulated annealing to a single problem formulation, we use simulated annealing as a problem formulation tool. In particular, we apply simulated annealing in the comparison of internal clustering criteria.

Careful selection of the clustering criterion is necessary because the "optimal" clustering for a particular criterion is not necessarily the "true" clustering; i.e., it might not represent the true underlying structure of the data. We use simulated annealing to find near-optimal clusterings for each of a set of criteria. By comparing these optimal clusterings with the true clusterings using an *external clustering criterion* (a criterion that uses information unavailable to the clustering algorithms' internal clustering criteria), we determine which internal criterion best approximates the true structure. Once an appropriate internal clustering criterion has been selected, one can construct a tailored clustering algorithm to solve the problem more efficiently. For example, if the "best" internal criterion in a clustering domain is "squared error," then the algorithm should be based on the location of cluster means; one might use one of the K-means algorithms in (Hartigan 1975). Using simulated annealing for criterion comparison provides some reassurance that the tailored algorithm is solving the "right" problem.

The method, called SImulation of Near-optima for Internal Clustering Criteria (SINICC, pronounced "cynic"), takes into account the effects of the parameters often used in internal criteria by allowing the user to specify and test a range of parameter values. SINICC works as follows:

1. Select a set S of M data sets representative of the problem domain.

2. Select a set Π_J of parameter values for each internal clustering criterion J.

3. Use simulated annealing to find near-optimal clusterings for each clustering criterion J with each parameter value in Π_J. Repeat for each data set in S.

4. Compare the near-optimal clusterings using an external clustering criterion.

Evaluating the criteria over a range of parameter values highlights any sensitivities of the criteria. It also allows the user to select the best parameter setting for future clustering applications in the problem domain.

The remainder of this paper roughly corresponds to the steps in the SINICC procedure. Section 2 describes partitional clustering as a combinatorial optimization problem. Section 3 focuses on simulated annealing as a method for finding near-optimal solutions to clustering problems. Section 4 builds on the previous sections to show how simulated annealing can be used in criterion comparison. Throughout the paper we use examples from a real clustering application in (Spillane et al. 1989), including a comparison of two internal clustering criteria.

2. The Clustering Problem

Although there is no fundamental clustering problem, some formulations are more general than others. This section first describes a very general formulation, then it details a special case that corresponds to a popular class of clustering algorithms. At a basic level, clustering is a combinatorial optimization problem:

Let

Q be the set containing all objects to be clustered,

C be the set of all feasible clusterings of Q,

$J: C \rightarrow \Re$ be the internal clustering criterion;

Then

Minimize $J(c)$ (1)

Subject To

$c \in C$. (2)

These two equations represent the most general form of the optimal clustering problem. The objective is to find the clustering c that minimizes an internal clustering criterion J. The set C defines c's data structure, including all the feasible clusterings of the set Q of all objects to be clustered.

A clustering algorithm maps Q into C. There are two basic types of clustering algorithms. The first type is partitional algorithms, which construct a simple partitioning of Q into a set of nonoverlapping clusters. The second type is hierarchical algorithms, which decompose Q into several levels of partitionings. Hierarchical decomposition is structured as a dendrogram, a tree that iteratively splits Q into smaller subsets until each object is in its own subset. The dendrogram can be created from the leaves up to the root (the "agglomerative" approach) or from the root down to the leaves (the "divisive" approach). The most common agglomerative clustering schemes are described in (Johnson 1967).

Partitioning is most appropriate when one is only interested in the subsets, while hierarchical decomposition is most applicable when one seeks to show similarity relationships between clusters. Section 2.1 formalizes the combinatorics of the partitional strategy. Although the simulated annealing algorithm described in section 3 is configurable for either partitional or the hierarchical clustering, the emphasis in this paper is on partitional formulations.

2.1 Partitional Clustering

Building on the basic combinatorial problem in (1) and (2), we define optimal partitioning, where the vector **p** represents the assignment of objects to clusters:

Let

Q be the set of all objects to be clustered,

$n = |Q|$ be the number of objects in Q,

$k \le n$ be the maximum number of clusters,

$P = \{\mathbf{p}: \forall i \in \{1,\ldots,n\}, p_i \in \{1,\ldots,k\}\}$ be the set of all partitionings,

$J: P \rightarrow \Re$ be the internal clustering criterion;

Then

Minimize $J(\mathbf{p})$ (3)

Subject to

$\mathbf{p} \in P$. (4)

Each cluster has a unique, integer cluster "label" in $\{1,...,k\}$, and the vector **p** assigns a cluster label p_i to the i-th object in Q. The function J maps elements of P into a real-valued cost. We formulate clustering as an assignment problem here to facilitate direct implementation of combinatorial optimization techniques (e.g., simulated annealing).

There are a variety of algorithms to solve such a problem. A thorough survey of partitional clustering algorithms is in (Jain and Dubes 1988). Few partitional algorithms guarantee a global-optimum solution to their associated problem formulation. K-means, for example, uses a greedy improvement heuristic to approximate the best "squared error" clustering. Thus, the algorithm is based on minimizing the total squared distance of the objects to their associated cluster means. There are many variants on K-means, and many of them converge rapidly on a locally optimal clustering, but none converge on the global optimum. As shown in (Klein and Dubes 1989), simulated annealing tends to find significantly better clusterings, but often requires much greater computational effort.

Unlike K-means or simulated annealing, some algorithms don't have any clear objective. Often, they solve a constraint-satisfaction problem. Consider, for example, ISODATA (Ball and Hall 1965), a popular partitioning algorithm based on a squared error criterion with $k=n$. Since minimizing squared error with $k=n$ is solved by placing each object in its own cluster, ISODATA translates this underlying objective into a set of

"splitting" and "lumping" constraints on the clusters. The algorithm starts with an arbitrary clustering and splits or joins clusters until all clusters satisfy the splitting and lumping constraints, settling on some number $k' \leq n$ of clusters. Although simulated annealing cannot be applied directly to constraint satisfaction problems, one can often define a J that approximates the meaning of one or more constraints. An example is shown in the next section.

2.2 An Example Problem Domain

Surveillance problems represent a relatively recent and important application domain for clustering algorithms. As a specific example of this class of problems, consider the monitoring of ground-based "entities" by airborne and ground-based sensors. The exact number and location of entities at any time are unknown. Each sensor generates reports about the entities within its range. A sensor report provides an estimated location for a entity and an elliptical error probable (a 95% Gaussian confidence region for the entity's location).

The sensors transmit their reports to a central processing center, which maintains a database describing the likely locations of sensed entities. Whenever reports are received from the sensors, the system compares them to previous reports. If it appears that an incoming report represents a new entity, then the system adds a record of this new entity to the database. On the other hand, if the sensor report appears to correspond to a previously observed entity, then the system updates the database record of this entity using information from the new sensor. The decision to make a new record or update an old record is known variously as data *correlation* (the name we use here) or data *association* (cf., (Spillane et al. 1989)). A separate problem not considered here, but also involved in this correlation processing, is the classification of the entity. For the purposes of this paper, we treat all entities as if they were from the same class.

The ability to manually correlate reports decreases as the number of reports arriving at the processing center increases. In fact, the number and variety of sensors has outstripped the capabilities of current manual processing stations to effectively monitor entity activity. Hence, there is widespread interest in automated techniques for correlation decision making.

If the reports are collected in batches and the correlation decision is made optimally with respect to some clustering criterion, then the data correlation problem reduces to

partitional clustering without a predetermined number of clusters. Brown et al. (1990) simulated this problem in order to evaluate various approaches to correlation. The simulation models the activity of a set of N entities distributed with uniform probability over a 120 by 80 kilometer grid. A set of airborne sensors report the entities at irregular intervals. Each report consists of a pair (X, Σ), where X is a two-dimensional location estimate and Σ is a 95% confidence ellipse about X. Hence, the objective is to cluster (or correlate) reports that pertain to the same entity.

Figure 1 shows a representative data set from Brown et al.'s simulation. Each point on the grid represents the X component from a sensor report. There are 194 such points in figure 1. The circles — each with a 5 kilometer (Euclidean) radius — show at their centers the means of the "true" clusters. In other words, if one actually knew which reports pertain to the same entity, then the circles circumscribe the best estimate for each entity's location. There are twenty such circles in the figure, corresponding to twenty actual entities present in the environment. Obviously, the true cluster means are unknown to the clustering algorithm, and are shown here to improve problem understanding and for comparison with figures 2 and 3 in the next section.

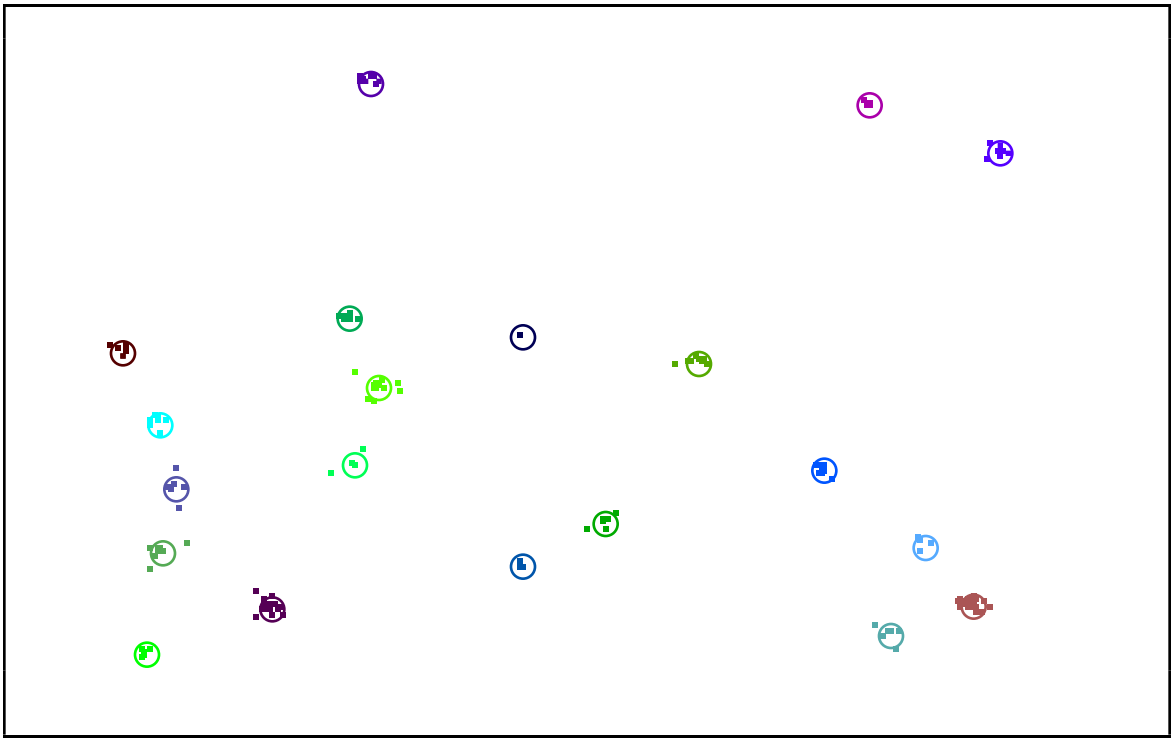

Figure 1. A Sample Data Set from the Simulation in (Spillane et al. 1989).

The choice of the criterion for use in this partitional clustering problem is critical to the correlation decision making process. For computational reasons, it is desired that the criterion be simple. One of the simplest of the internal clustering criteria is total within-cluster distance:

$$W(\mathbf{p}) = \sum_{p_i = p_j} d_{ij} \tag{11}$$

This criterion is simpler than squared error (and other, similar criteria) because all distance calculations can be preprocessed and stored in a static matrix. However, since minimizing W(**p**) with $k=n$ places each report in its own cluster, the user must estimate the true number of clusters, which is non-trivial for many real surveillance situations because the number of entities sensed often varies over time. So, although W(**p**) is simple enough for straightforward application of most combinatorial optimization algorithms (e.g., integer programming techniques) to small problem instances, applicability is limited to cases in which one can accurately estimate the true number of clusters.

Barker (1989) eliminated the need to accurately estimate the number of clusters by incorporating a distance threshold v into W(**p**):

$$B(\mathbf{p}) = \sum_{p_i = p_j} (d_{ij} - v) \tag{12}$$

Barker's new formulation, which is computationally identical to W(**p**) once all of the (d_{ij}-v) terms have been preprocessed, is analogous to the constraint satisfaction problem solved by ISODATA. The criterion penalizes large clusters by adding in more d_{ij} values. It penalizes small clusters by subtracting fewer v values. Hence, Barker's formulation captures the spirit of the two ISODATA constraints with penalties for large and small clusters. There is a practical advantage to Barker's formulation, however, because it makes the tradeoff between large and small clusters explicit.

Barker's criterion and total within-cluster distance are competing criteria for use in the data correlation problem and similar clustering problems. Obviously a system designer would like information on the performance of criteria such as these before they are implemented in an report processing center. The next section describes simulated annealing for clustering and shows how it can be used as the basis for evaluating criteria over a specified problem domain.

3. Simulated Annealing for Clustering

3.1 The General Algorithm

Simulated annealing (SA) is a powerful optimization technique that attempts to find a global minimum of a function using concepts borrowed from Statistical Mechanics. Although it was first described in its entirety by (Kirkpatrick et al. 1983), significant portions of the method were known as early as 1953 (Metropolis et al. 1953).

The algorithm described by Metropolis et al. is the heart of any classical implementation of SA. The algorithm was originally intended for simulating the evolution of a solid in a heat bath to thermal equilibrium. As it was first described, the algorithm starts with a "substance" composed of many interacting individual molecules arranged in a random fashion. Then, small random perturbations to the structure of the molecules are attempted, and each perturbation is accepted with a probability based on the associated "energy" increase, ΔE. If ΔE is at least 0, then the perturbation is accepted with probability $\exp(\Delta E/T)$, where T is the "temperature" of the substance. If ΔE is less than 0, then the perturbation is accepted with probability 1. Eventually, after a large number of trial perturbations, the energy settles to an equilibrium appropriate for the temperature. At high temperatures, the value of $\exp(\Delta E/T)$ is close to 1, regardless of the increase in energy, meaning that almost all perturbations are accepted and the resulting structures are very random. Thus, the algorithm at high temperature does not settle on any particular structure, regardless of the initial arrangement. At low temperatures, however, the process exhibits a significant bias towards perturbations that cause energy decreases. Eventually, the Metropolis algorithm at low temperature settles on a structure that has low energy, but the structure depends highly on the initial arrangement. SA uses both the high- and the low-temperature properties of the Metropolis algorithm to find low energy, irregardless of the intial structure.

In metallurgy, the minimum energy state is often sought using "process annealing," in which the substance is initially heated to a very high temperature and then slowly cooled to room temperature. The heating process allows a very stable, sub-optimal structure to be relaxed to a more pliable, less-stressed (i.e., low-energy) structure before cooling. The cooling is made slow to overcome the high dependence of low temperature equilibrium energies on the initial state. If the cooling is too fast (a process called "quenching") then the resulting structure is likely to be sub-optimal.

Simulated annealing exploits the obvious analogy between process annealing and combinatorial optimization problems, where the "molecules" are the variables in the data structure and the "energy" function is the objective function. Algorithm 1 shows simulated annealing in the context of the general combinatorial optimization problem in equations 1 and 2. In the combinatorial optimization framework, the "temperature" is a real-valued scalar that controls the degree of randomness of the search. At high temperatures, the algorithm behaves like random search. At low temperatures, it behaves like greedy local search. SA slowly decreases the temperature (by a factor of α each iteration) from the initial temperature T_0 to the final temperature T_f, by which time the values of the decision variables have "frozen" into a very stable state. As shown in (Aarts and Korst 1989, pg.17), the limiting state as the temperature approaches zero is the global minimum.

Procedure SA(δ,MaxIt,T_0,α,T_f)

Let C be the set of all feasible clusterings,
$c, c' \in C$ be the current and perturbed clusterings, repectively,
$\delta : C \to C$ be a randomized perturbation operator,
$J: C \to \Re^+$ be the internal clustering criterion,
$T \in \Re^+$ be a "temperature" parameter that controls the "greediness",
$U: \Im^2 \to [0,1)$ be a function that returns a random number between 0 and 1,
$MaxIt \in \Im^+$ be the number of iterations of the Metropolis algorithm,
$\alpha \in \Re^+$, $\alpha < 1$ be an "attenuation" constant for reducing the temperature,
T_0 and T_f be the initial and final temperatures.

$T \leftarrow T_0$
REPEAT
  FOR $i \leftarrow 1$ TO MaxIt DO
    $c' \leftarrow \delta(c)$
    $\Delta \leftarrow J(c') - J(c)$
    IF $\Delta < 0$ OR ($e^{-\Delta / T} \geq U[0,1]$) THEN
      $c \leftarrow c'$
  ENDFOR
  $T \leftarrow \alpha T$
UNTIL $T \leq T_f$

Algorithm 1. Simulated Annealing for Clustering.

3.2 SA for Partitional Clustering

The application of simulated annealing to the partitional clustering formulation in equations 3 and 4 is straightforward. This section describes two remaining problem-dependent details: the perturbation operator δ and the annealing schedule $(\text{MaxIt}, T_0, \alpha, T_f)$ for partitional clustering. In addition, we briefly suggest how one might apply simulated annealing to hierarchical clustering.

The perturbation operator for partitional clustering switches a randomly-chosen object i in Q from one cluster to another randomly chosen cluster. Algorithm 2 shows the basic procedure. The set L contains the cluster labels used in $\mathbf{p}$. Similarly, L^c contains the labels not used in $\mathbf{p}$. The switching procedure first selects an integer m in the range $[0, |L|]$. If m equals 0 and there exists an unused cluster label (i.e., $|L|<k$), then object i is placed in its own singleton cluster. Otherwise, i switches to another, existing cluster.

FUNCTION $\delta(\mathbf{p})$

Let $n = |Q|$ be the number of objects to be clustered,
$L = \{i \in \{1,\ldots,k\}: \exists m \in \{1,\ldots,n\} \ni p_m = i\}$ be the set cluster labels in $\mathbf{p}$,
$L^c = \{i \in \{1,\ldots,k\}: i \notin L\}$ be the set of cluster labels unused in $\mathbf{p}$,
SELECT(*range*) be a function that returns a random element from the set *range*,
$\mathbf{p}, \mathbf{p}' \in P$ be the original and perturbed partitionings, respectively.

$\mathbf{p}' \leftarrow \mathbf{p}$
$i \leftarrow$ SELECT(1,...,n)
REPEAT
    $m \leftarrow$ SELECT(0,...,|L|)
    IF $|L| = k$ OR $m > 0$ THEN
        $p_i' \leftarrow$ SELECT(L)
    ELSE
        $p_i' \leftarrow$ SELECT(L^c)
    ENDELSE
UNTIL $p_i' \neq p_i$
RETURN $\mathbf{p}'$

Algorithm 2. A Perturbation Operator for Partitional Clustering.

Since SA is used here for comparison purposes, we have designed the annealing schedule to standardize the computational effort without compromising the quality of the resulting clustering solutions. The computational effort is made fair by allowing each run a fixed number of trial perturbations. The total number of pertubations tried in any run is MaxIt·NumTemp, where MaxIt is a fixed multiple of the number of objects to be clustered and NumTemp is a user-defined constant. The solution is made accurate using a very conservative annealing schedule. We calculate the initial temperature with the formula from (Aarts and van Laarhoven 1985), which uses statistics compiled from MaxIt random permutations:

$$T_0 = \mu^+ \Big/ \log\left(\frac{m^+}{\chi m^+ - (1-\chi)(\text{MaxIt} - m^+)}\right), \tag{13}$$

where

m^+ = the number of cost increases in MaxIt random perturbations,

μ^+ = the average cost increase over the perturbations,

χ = the acceptance ratio, a real - valued scalar in (0,1).

For the final temperature, we require that

$$e^{-\beta\mu^+/T_f} = \varepsilon, \qquad \text{where } 0 < \varepsilon < 1 \text{ and } 0 < \beta < 1, \tag{14}$$

meaning that at the final temperature SA accepts a cost increase of $\beta\mu^+$ with probability ε. This simplifies to the following:

$$T_f = -\beta\mu^+ / \log(\varepsilon). \tag{15}$$

This formula is analogous to the estimate in (White 1984), with $\beta\mu^+$ representing the smallest cost increase caused by a perturbation from a local minimum and ε^{-1} representing the number of perturbations possible at each step. With this interpretation, equation 15 implies that ε approximates the probability of escaping a local minimum at the final temperature. Given NumTemp, T_0, and T_f, the calculation of α is straightforward:

$$\alpha = \left(\frac{T_f}{T_0}\right)^{1/\text{NumTemp}}. \tag{16}$$

With sufficiently large NumTemp and MaxIt and sufficiently small $(1-\chi)$, β, and ε, the annealing schedule ensures slow, steady convergence to a near-global optimum clustering. For the runs reported in this paper, the settings were NumTemp=200, MaxIt=4*n*, $(1-\chi)$=0.25, β=0.125, and ε=0.00000000001.

Simulated annealing is not restricted to partitional clustering, but the implementation is not so straightforward for hierarchical clustering, where the complexity of the data structure complicates the definition of an appropriate perturbation operator. One possible perturbation operator is described in (Wallace and Kanade 1990); the operator is too complex for description here.

3.4 Application to the Example Data

This section informally compares the near-optimal clusterings for the W(**p**) and B(**p**) criteria in equations (11) and (12). Figures 2 and 3 show simulated annealing converging on a near-optimal clustering for each formulation when applied to the data in figure 1. (The parameter values, k=20 for W(**p**) and v=3.5 for B(**p**), are the best found in the testing described in section 4.2.) Each figure shows a chronological sequence of four graphic screens from our Apple Macintosh IIx implementation, written in C. (We also implemented a text-based version on an Intel i860 hypercube.) The graphical version displays each report as a point on a 640x200 grid, where each point's color represents its cluster; one can watch the clusterings converge by noting the color changes on the screen. (Although the computer implementation is in color, the figures are in black and white to facilitate display in print. As in figure 1, the circles denote the current cluster means.) The first screen shows the clustering a few seconds into the run. The second screen shows it exactly one-quarter of the way through the run. The third is exactly half-way through the run. The last screen shows the final clustering.

Closer examination of figures 2 and 3 highlight characteristics of the criteria that might go unnoticed in a purely statistical comparison. For example, in figure 2 we see that large variances in point density can fool W(**p**). Consider the three clumps of points at the lower right-hand corner of the sample, shown in detail in figure 4. The rightmost of the three clumps contains 19 points while the other two have a total of ten points. There are 171 interpoint distances in the rightmost clump that can contribute to W(**p**), while the other two clumps combined can only contribute up to 45 distances. This means that although the distances within the rightmost clump are relatively small, their contribution to W(**p**) can dominate that of the points in the other two clumps. Hence, SA with W(**p**) split the rightmost clump into two dense clusters and combined the other two clumps into one sparse cluster. So, point density must be considered when using W(**p**).

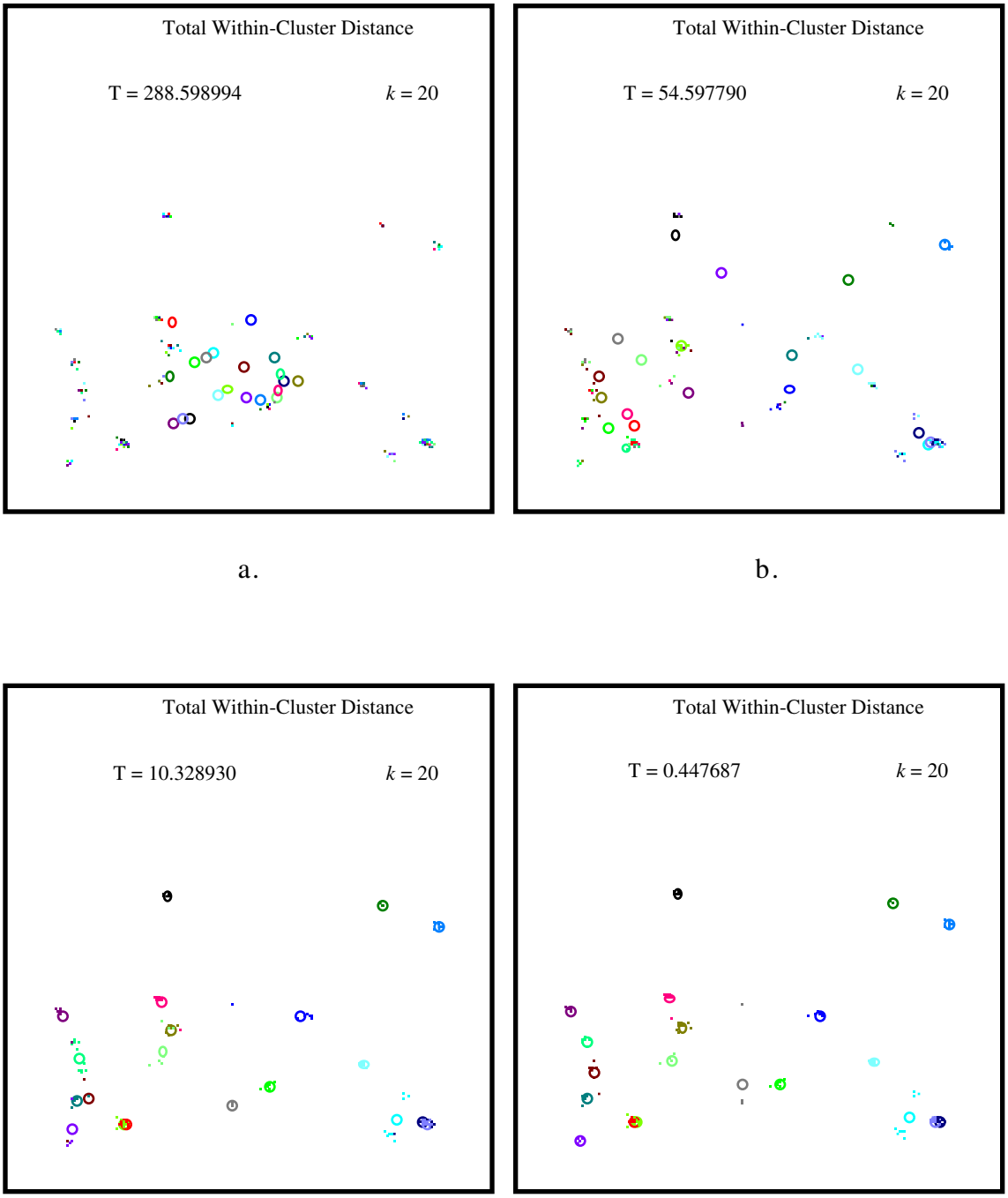

Figure 2. Convergence of Simulated Annealing for Total Within-Cluster Distance.

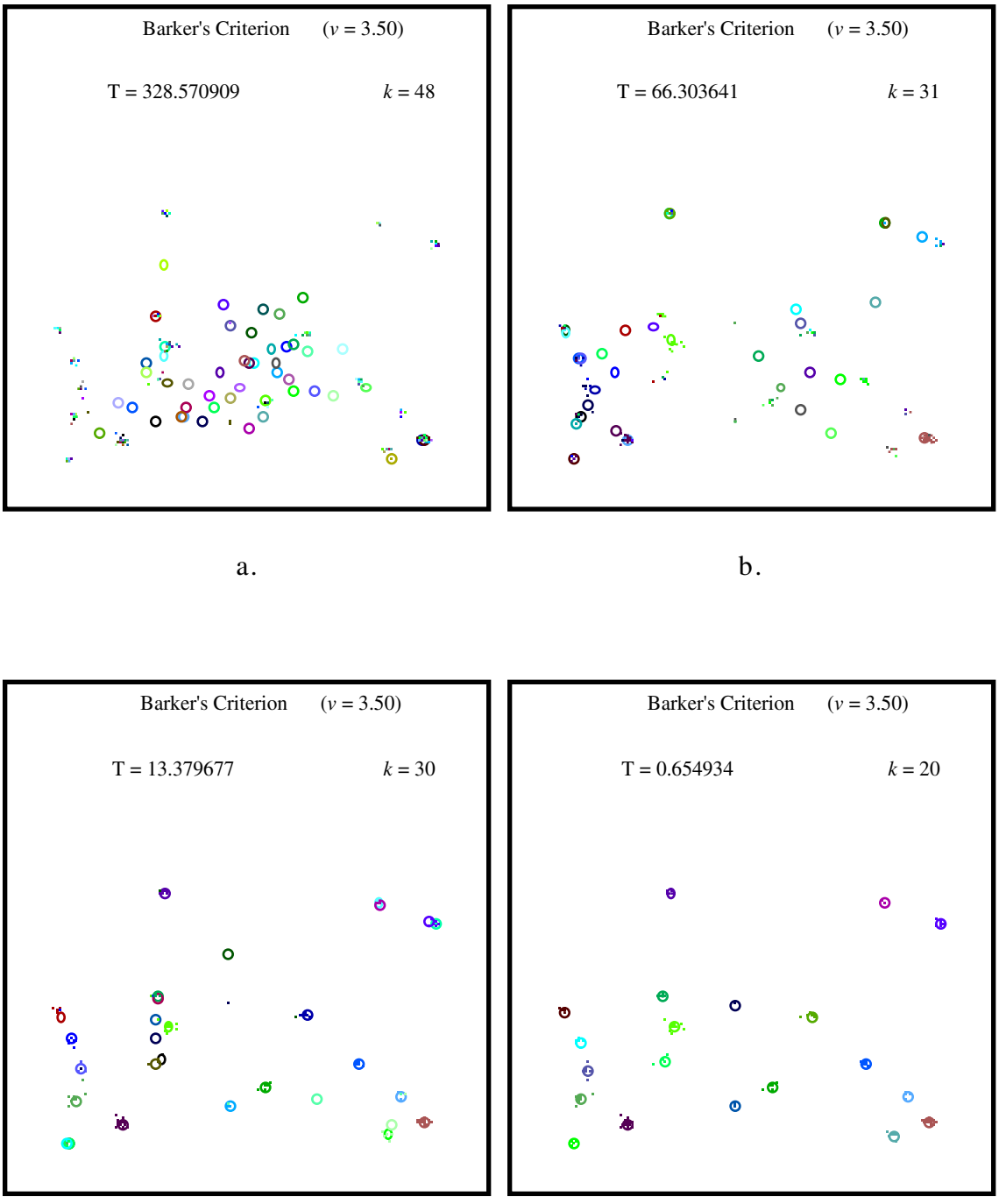

Figure 3. Convergence of Simulated Annealing for Barker's Criterion (Barker 1989).

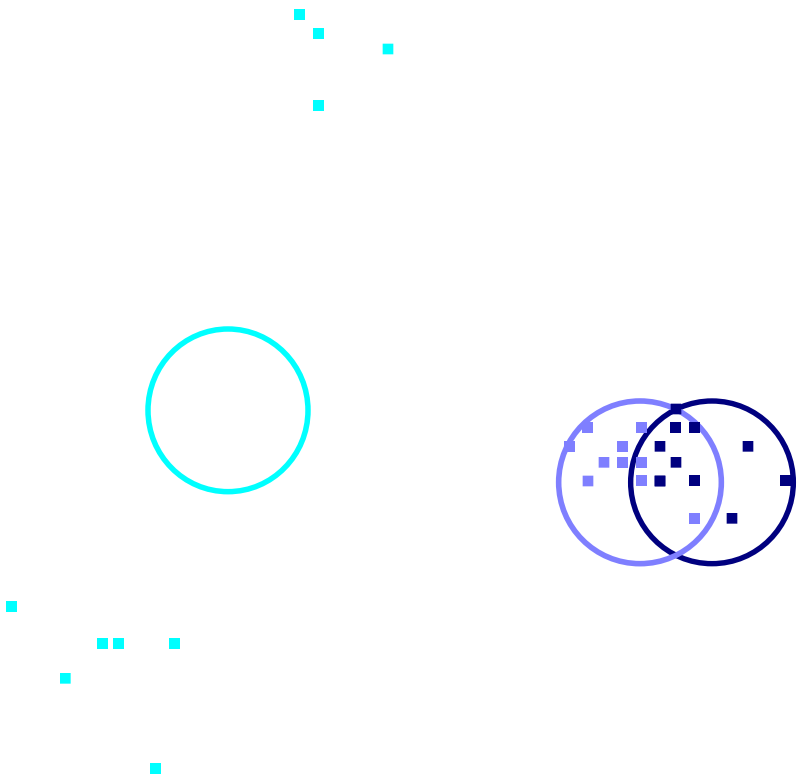

Figure 4. Detail of the Lower Right-Hand Side of Figure 2d.

Although B(**p**) did not split the dense clusters as often, it occasionally had some problems of its own. Figure 5 demonstrates such a case, extracted from a data set that presented problems for both W(**p**) and B(**p**), even when using their "best" parameter values. Figure 5a shows the true clustering of two overlapping groups of points. In figure 5b, which shows the B(**p**) clustering, the densest parts of the two groups merge to form a large, dense cluster flanked by two singleton clusters. Since B(**p**) was designed to seek out areas with high point density, it tends to cluster points in the high density regions, even if it means creating an extra cluster or two. In the example, there were just enough points within v kilometers of each other to force clustering on the overlap between the two true clusters. As shown by figure 5c, although W(**p**) was not perfect, it was better than B(**p**) in this special case.

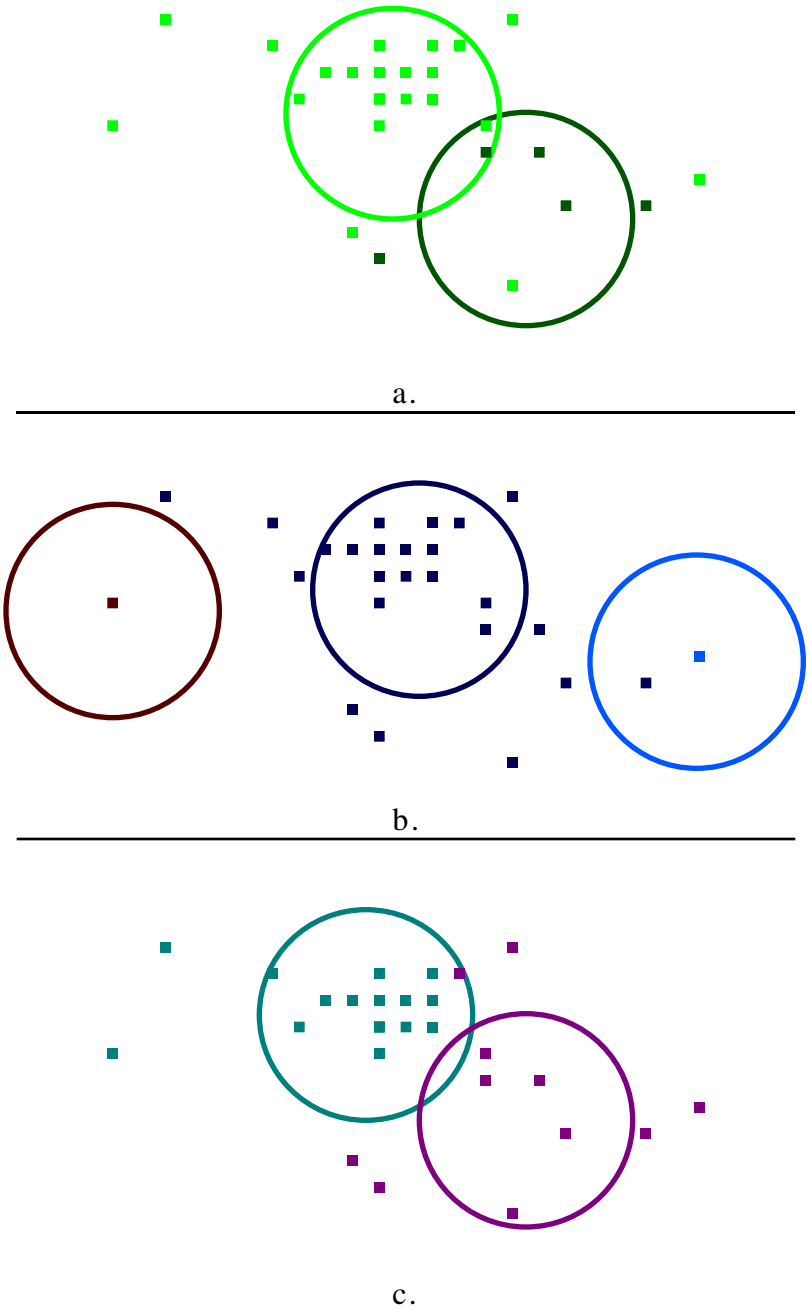

Figure 5. Detail From a Data Set with Overlapping Clusters for (a) the True Clustering, (b) the B(**p**) Clustering, and (c) the W(**p**) Clustering.

4. Application of SINICC to the Problem Domain

This section reports the results from using W(**p**) and B(**p**) in the sample problem domain. Following the SINICC procedure described in the introduction, we first describe the external clustering criteria used: the Rand statistic (Rand 1971) and the Jaccard statistic in (Anderberg 1973). Then we present a statistical analysis based on these criteria for thirty-two data sets generated by the simulation in (Brown et al. 1990). By comparing the results from SA with each criterion over a range of parameter values, we formalize the tendencies observed in the visual analysis.

4.1 External Clustering Criteria

Both the Rand and the Jaccard criteria require that one know which objects truly cluster together (i.e., one needs to know the true partitioning). Using the notation in section 2.1, the criteria measure the similarity between the true partitioning **g** and the partitioning **p** returned by a clustering method. Both measures use the following statistics:

$$s_+^+ = \quad \text{The number of times that } g_i = g_j \text{ when } p_i = p_j, \tag{17}$$

$$s_-^- = \quad \text{The number of times that } g_i \neq g_j \text{ when } p_i \neq p_j, \tag{18}$$

$$s_-^+ = \quad \text{The number of times that } g_i = g_j \text{ when } p_i \neq p_j, \text{ and} \tag{19}$$

$$s_+^- = \quad \text{The number of times that } g_i \neq g_j \text{ when } p_i = p_j. \tag{20}$$

The first two statistics count the number of time that **p** agrees with **g**, while the last two count the number of disagreements. The Rand criterion calculates the ratio of agreements to the total number of comparisons:

$$\text{Rand}(\mathbf{g},\mathbf{p}) = \frac{s_+^+ + s_-^-}{s_+^+ + s_-^- + s_-^+ + s_+^-} = \frac{s_+^+ + s_-^-}{n(n-1)/2} \tag{21}$$

The Jaccard criterion is calculated similarly, except for the omission of the (negative-negative) agreement statistic:

$$\text{Jaccard}(\mathbf{g},\mathbf{p}) = \frac{s_+^+}{s_+^+ + s_-^+ + s_+^-} \tag{22}$$

Because Jaccard's measure is monotonic with Rand's, improvement in one of the measures implies improvement in the other. Therefore, since the Jaccard criterion is more sensitive than the Rand criterion, we only report the Jaccard score in the next section.

4.2 Test Results

In testing on Oak Ridge National Laboratory's Intel i860 hypercube, we compared the best test results for W(**p**) and B(**p**) over 32 data sets, where each data set has between 150 and 600 objects, split into 20 true clusters on the average. Table 1 shows for each data set the best Jaccard score for each of the criteria. In addition, for W(**p**) it shows the best "number of clusters" parameter k in {18, 19, 20, 21, 22}. Similarly, for B(**p**) the table shows the best "median distance" parameter v in the set {2.0, 2.5, 3.0, 3.5, 4.0} and the associated number of clusters in the final partitioning. At the bottom of the table is the minimum, maximum, and mean of each column.

Tables 2 and 3 may be of use for practitioners, who do not have the benefit of knowing the best parameter values for a given data set. For each of the parameter values tested, the tables show the average performance of the criteria over the 32 data sets. This allows the user to select the "best" default parameter value in a given range.

From table 2, it appears that a good initial estimate for the number of clusters k in a future data set is 18, for which W(**p**)'s average Jaccard score is approximately 72%. Since the best Jaccard scores were achieved with k=18 (whereas the mean number of true clusters is 20), it is likely that Jaccard-optimal W(**p**) clusterings underestimate k by at least 2 clusters.

From table 3, good estimates for v are in the range [3.0, 3.5], for which B(**p**)'s average Jaccard score is approximately 95%. Note also that the B(p)'s worst performance over the rnage [2.0, 5.0] is 86%. Hence, even if the best v for a particular data set is not in the range, [3.0, 3.5], B(**p**) still seems to outperform W(**p**).

	W(**p**)		B(**p**)		
Data Set	**Best Jaccard**	**Parameter** *k*	**Best Jaccard**	**Parameter** *v*	**Resulting** *k*
1	0.712	18	0.932	2.0	28
2	0.778	18	1.000	4.0	20
3	0.624	21	0.985	2.5	23
4	0.705	20	0.989	3.5	21
5	0.748	21	0.959	3.0	22
6	0.698	19	0.894	2.5	27
7	0.622	19	0.952	3.0	20
8	0.725	18	1.000	3.5	20
9	0.641	19	0.807	3.0	21
10	0.812	18	0.992	3.5	21
11	0.611	19	1.000	3.0	21
12	0.682	20	0.928	3.5	18
13	0.749	20	0.988	3.0	21
14	0.859	18	1.000	3.5	20
15	0.589	20	1.000	3.5	21
16	0.978	19	0.936	2.5	24
17	0.807	19	0.981	3.0	22
18	0.797	19	1.000	3.0	21
19	0.773	18	1.000	3.5	20
20	0.903	19	0.949	3.5	19
21	0.831	18	0.992	3.0	22
22	0.733	18	0.953	3.0	20
23	0.792	20	0.957	3.0	23
24	0.683	18	0.935	3.0	21
25	0.652	18	0.851	4.0	18
26	0.719	19	1.000	3.0	21
27	0.916	18	1.000	3.5	20
28	0.674	18	0.935	2.5	24
29	0.726	19	0.990	3.5	19
30	0.815	18	1.000	3.5	20
31	0.774	19	0.992	3.5	21
32	0.793	19	0.939	3.0	23
Min	0.589	18.0	0.807	2.0	18.0
Avg	**0.747**	**18.9**	**0.963**	**3.2**	**21.3**
Max	0.978	21.0	1.000	4.0	28.0

Table 1. The Best Results for Each of 32 Data Sets

	Parameter *k*				
	18	19	20	21	22
Avg. Score	0.726	0.705	0.684	0.655	0.620

Table 2. Average Jaccard Score by Parameter *k* for W(**p**).

	Parameter *v*								
	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
Avg Score	0.636	0.771	0.879	0.937	0.948	0.944	0.916	0.893	0.867

Table 3. Average Jaccard Score by Parameter *v* for B(**p**).

5. Conclusion

Previous studies found simulated annealing to be impractical for clustering. The results in this paper show that simulated annealing is an effective search procedure for use in evaluating clustering criteria. The evaluation of clustering criteria is problematic. If two criteria are compared using a stepwise (greedy) procedure, then the comparison is as much a function of the local optima found by the greedy procedure as it is of the criteria themselves. Simulated annealing can find near optimal clusterings for each each of the evaluated criteria; thus, simulated annealing provides a fairer basis for comparing criteria.

Our method of criteria evaluation, SINICC, uses simulated to find near optimal clusterings over a range of user specified parameter values. Testing over a range of parameters shows the sensitivity of the criteria to parameter settings. SINICC also allows for testing over a variety of data sets, and scores performance with an external clustering criterion. Extensive testing is possible because of SINICC's implementation on a multi-processor (Intel i860 hypercube).

We applied SINICC to evaluate two criteria, within-cluster distance and Barker's criterion, for a surveillance problem. The surveillance simulation modeled the activities of airborne sensors operating against ground targets that move within a 120 by 80 km. area. The problem is to cluster sensor reports so that each cluster represents a single entity.

Results from our testing showed that Barker's criterion outperformed within-cluster distance. In fact, the worst Jaccard score for Barker's criterion was better than the average Jaccard score for within-cluster distance. In addition to obtaining raw scores for the criteria, SINICC allowed us to view the clusterings with each criterion. As a result we were able to detect problems the within-cluster distance criterion has with dense clusters (figure 4). SINICC also allowed us to view the tendency of Barker's criterion to add clusters when two entities were very close.

This work provides clear evidence that simulated annealing is useful for criteria evaluation with partitional clustering methods. Additional work can extend the use of simulated annealing to comparisons of hierarchical approaches. New perturbation operators will be needed to implement this extension. Simulated annealing also represents a promising approach to mixture model methods for clustering. Hence, while simulated annealing might not be appropriate for direct applications of clustering methods, it does represent a practical tool for the evaluation and refinement of clustering techniques.

6. References

Aarts, E., and J. Korst. 1989. *Simulated Annealing and Boltzmann Machines*. Wiley, New York.

Aarts, E. H. L., and P. J. M. van Laarhoven. 1985. A New Polynomial Time Cooling Schedule. In *Proceedings of the IEEE International Conference on Computer-Aided Design*, 206-208, Santa Clara, CA.

Anderberg, M. 1973. *Cluster Analysis for Applications*. Academic Press, New York, NY.

Ball, G., and D. Hall. 1965. ISODATA, a Novel Method of Data Analysis and Classification. Research Report AD-699616, Stanford Research Institute, Stanford, CA.

Barker, A. 1989. *Neural Networks for Data Fusion*. Masters Thesis, University of Virginia, Charlottesville, VA.

Brown, D., C. Pittard, and A. Spillane. 1990. ASSET: A Simulation Test Bed for Data Association Algorithms. Research Report IPC-TR-90-002, University of Virginia, Charlottesville, VA.

Hartigan, J. 1975. *Clustering Algorithms*. Wiley, New York, NY.

Jain, A., and R. Dubes. 1988. *Algorithms for Clustering data*. Prentice Hall, Englewood Cliffs, NJ.

Johnson, S. 1967. Hierarchical Clustering Schemes. *Psychometrika* **32**, 241-254.

Kirkpatrick, S., C. Gelatt, and M. Vecchi. 1983. Optimization by Simulated Annealing. *Science* **220**, 671-680.

Klein, R., and R. Dubes. 1989. Experiments in Projection and Clustering by Simulated Annealing. *Pattern Recognition* **22**, 213-220.

Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. 1953. Equations of State Calculations by Fast Computing Machines. *Journal of Chemical Physics* **21**, 1087-1092.

Rand, W. 1971. Objective Criteria for the Evaluation of Clustering Algorithms . *Journal of the American Statistical Association* **66**, 846-850.

Spillane, A., D. Brown, and C. Pittard. 1989. A Method for the Evaluation of Correlation Algorithms. Research Report IPC-TR-89-004, University of Virginia, Charlottesville, VA.

Wallace, R., and T. Kanade. 1990. Finding Natural Clusters Having Minimal Description Length. In *IEEE*, 438-442.

White, S. R. 1984. Concepts of Scale in Simulated Annealing. In *Proceedings of the IEEE International Conference on Computer Design*, 646-651, Port Chester.