Hindawi Publishing Corporation
ISRN Computational Mathematics
Volume 2013, Article ID 931019, 8 pages
http://dx.doi.org/10.1155/2013/931019

Research Article

Evolutionary Algorithms for Robust Density-Based Data Clustering

Amit Banerjee

School of Science, Engineering and Technology, Penn State Harrisburg, Middletown, PA 17057, USA

Correspondence should be addressed to Amit Banerjee; aub25@psu.edu

Received 28 November 2012; Accepted 17 December 2012

Academic Editors: R. A. Krohling and R. Pandey

Density-based clustering methods are known to be robust against outliers in data; however, they are sensitive to user-specified parameters, the selection of which is not trivial. Moreover, relational data clustering is an area that has received considerably less attention than object data clustering. In this paper, two approaches to robust density-based clustering for relational data using evolutionary computation are investigated.

1. Introduction

Clustering as an integral machine learning activity involves unsupervised classification of data into self-similar clusters—entities in a cluster are alike and entities across clusters are not. A cluster is defined in terms of internal homogeneity and external separation, or in density-related terms, clusters are dense regions in feature space with sparser regions separating clusters from one another. Datasets themselves can be divided into two groups—object data and relational data; the distinction that is described later in the paper. While a lot of effort and research have gone into developing clustering algorithms for object data, data clustering methods for relational data have received lesser attention. In application domains such as social sciences and bioinformatics, relational datasets are more common than object data. Prototype-based clustering algorithms are popular for clustering object data, where a cluster can be represented by a cluster prototype, and algorithms are built around optimization parameters of the prototypes. Most optimization is done iteratively from a randomly chosen initial state, and as it turns out, prototype-based object clustering is very sensitive to this initialization. Evolutionary algorithms and other approaches that operate on a population of potential solutions have lately been used as a remedy to the *curse of initialization*. Real-life data is also inherently noisy, and prototype-based clustering methods have been shown to be adversely affected by noise in data. Unless guarded against, the presence of outliers in data influences the calculation of prototype parameters. Density-based clustering algorithms are resistant to outliers if it can be assumed that outliers occupy the less-dense regions in the feature space. Density-based spatial clustering of applications with noise (DBSCAN) is the most popular density-based clustering algorithm [1]. Resistant to outliers and easily adapted to large-scale data clustering, DBSCAN and its variants still suffer from the problem of prespecification of two important parameters, described later, which in practice is not always straightforward or trivial. In this paper, two evolutionary approaches to relational data clustering are presented—one that converts relational data to an equivalent object data and simultaneously partitions the data, and another that implements a relational noise-resistant version of the DBSCAN.

2. Relational Data Clustering

A set of entities $O = \{o_1, o_2, \ldots, o_n\}$, $n \in \mathbb{N}$, can be numerically specified by an object dataset $X = \{x_1, x_2, \ldots, x_n\} \in \mathbb{R}^d$, $d \in \mathbb{N}$, where each of the n entities is numerically described by d real-valued features. In particular, discrete features are included in this definition; however, object data can consist of nominal (categorical) or ordinal values, interval or ratio values, or can be purely descriptive or consist of a combination of features. Each object is represented as a point in the d-dimensional feature space, so an object dataset can

be visualized by plotting it in the feature space. The same data can also be numerically specified by a relational dataset $R \subset \mathbb{R}^{n\times n}$, where a pair of objects is numerically described by a real-valued relationship. Each entry r_{jk} $(1 \le j, k \le n)$ is a quantification of the relationship between objects o_j and o_k. Dissimilarity and similarity are common relationships; when objects can be numerically specified by X, distance or a function thereof can be used as a measure of dissimilarity or similarity. Positive relations such that $r_{jk} \ge 0$ and symmetric relations such that $r_{jk} = r_{kj}$ are often considered. Also similarity relations are required to be reflexive, that is, $r_{jj} = 1$, and dissimilarity relations are required to be irreflexive, that is, $r_{jj} = 0$ for all $1 \le j \le n$.

More often than not, relational data do not have an object basis and are just a quantification of how two entities are related to each other. Prototype-based clustering models that operate on object data often iteratively refine cluster assignments starting from an initial state by minimizing some distance measure of a cluster representative (called a prototype) from entities in that cluster. Examples of such methods include k-means clustering and fuzzy clustering. However, cluster prototypes are not well-defined in relational data, and hence prototype-based clustering models that have been reformulated for relational data compute the equivalent object data either implicitly or explicitly. Such methods include relational k-medoids, relational fuzzy c-means, fuzzy relational clustering and relational alternating cluster estimation [2–4]. In the absence of cluster prototypes, the methods are pseudoalternating at the very best, and cluster assignment is therefore computationally expensive. Another approach is data transformation which involves the linear or the nonlinear projection of the relational data into a lower dimensional object space, followed by prototype-based clustering in the transformed feature space. Since data clustering is the primary focus, simple transformations which roughly preserve the original interrelations between entities in a smaller dimensional feature space are preferred. Fuzzy nonlinear projection is a clustering method that simultaneously transforms and partitions the data. The method and its shortcomings are presented in detail in the next section.

3. Evolutionary Clustering

Clustering can be considered as a particular class of NP-hard problems [5], which has led to the development of efficient metaheuristics to provide optimal solutions to the clustering problem in reasonable time. The propensity of simple local search techniques like hill climbers and k-means to get trapped in local minima has also led to development of algorithms that search for multiple solutions such as ones based on evolutionary algorithms. The aim of evolutionary clustering algorithms is to evolve a population of (random) suboptimal partitions to a population that contains potential near-optimal partition(s). For more details, the reader is referred to Hruschka et al.'s review paper [6] on clustering using evolutionary algorithms.

One of the reasons why relational data clustering has seen fewer evolutionary algorithm-based implementation methodologies than object data clustering is because of the difficulty in coming up with a succinct yet meaningful genetic representation of the partition. Object data clustering using prototype methods is easier to implement using evolutionary algorithms, and most implementations seem to prefer the prototype-based string representation; that is, a k-partition of n data objects in d-dimensions can be represented by $k \times d$ string. This representation is preferred over the n-bit long cluster label representation since $k, d \ll n$. In the absence of explicit prototypes, certain judiciously chosen encodable parameters that are representative of the partition can be used to represent the partition in addition to the n-bit cluster-label (assignment) representation. Clustering algorithms such as DBSCAN produce uniquely different partitions that are a direct function of certain clustering parameters; however, prototype-based clustering methods are more sensitive to initialization and the number of clusters which cannot be considered clustering parameters. The drawback of using an n-bit cluster-label representation for relational data is that it cannot be directly and efficiently mapped on to the clustering criterion. In this paper, modified versions of both representations will be used.

For robust clustering, the challenge is to devise a suitable fitness function that will quantify the quality of a partition in the presence of outliers in the data and one that will reach its maximum when all the outliers have been correctly identified. One of the first robust evolutionary algorithms to be proposed used a function of the least median of squares (LMS) criterion as the fitness function to drive the evolution [7]. In fact, most evolutionary clustering algorithms use some kind of partition-dependent measure of homogeneity and separation as the fitness function.

Another strategy is to cast the robust clustering problem as a multiobjective optimization problem and solve using multi-objective evolutionary algorithms. The fitness function has to be designed as a pair (or more) of complementary functions in a way that Pareto-optimal fronts in the objective plane (or space if there are more than two objectives) can be extracted. The optimal solution to the robust clustering problem will then be found in the extreme Pareto-optimal front. The challenge lies in designing a suitable fitness function and devising a representation that will encode information about the objectives separately. The binary representation (0 for noise points, 1 for good points) concatenated with k-prototype locations is an easy representation to manipulate using canonical genetic operators, yet powerful enough to encode partition information in the nonnoisy subset of the data [8, 9]. Complementary objectives such as minimizing number of clusters along with uncovering dense regions of the data (the nonnoisy subset) and maximizing intercluster distances have been used successfully.

4. Relational Clustering Using Sammon Mapping

An indirect approach to clustering relational data is applying Sammon mapping to obtain an object-equivalent data in lower dimensions, followed by subjecting it to prototype-based clustering. Since the primary focus is clustering of data and not mapping accuracy, one can in principle map the

relational data into a 1-D data in a very limited range. This indirect approach is pursued in this paper. Sammon mapping produces an object dataset $Y = \{y_1, y_2, \ldots, y_n\} \in \mathbb{R}^q$, $q \in \mathbb{N}$ such that the distances $d_{jk} = \|y_j - y_k\|$, where $1 \leq j, k \leq n$ are as close as possible to the corresponding relational distance r_{jk}, that is, $\|d_{jk} - r_{jk}\| \approx 0$. If R is computed from an object data set $X \in \mathbb{R}^q$, then $X \approx Y$. Sammon mapping can be minimized by the error functional

$$J_1 = \frac{\sum_{j=1}^{n} \sum_{k=j+1}^{n} \left(\left(d_{jk} - r_{jk} \right)^2 / r_{jk} \right)}{\sum_{j=1}^{n} \sum_{k=j+1}^{n} r_{jk}}. \quad (1)$$

The minimization is performed using numerical optimization schemes such as gradient descent or Newton's algorithm. Sammon mapping has been applied to a relational data, followed by fuzzy clustering on the resulting Y [10], and Sammon mapping error functional has been explicitly incorporated into a clustering criteria (a more direct approach to clustering based on Sammon mapping) in fuzzy nonlinear projection algorithm [11]. In this paper, an optimization approach is proposed where the mapping error functional and a separate clustering criterion are simultaneously optimized.

Clustering criteria are used to evaluate individual partitions and are in essence a measure that maximizes both the homogeneity within clusters and the separation between clusters. Such measures like Davies-Bouldin Rule Index [12] are easily defined in the object space. For each cluster, a centroid is defined in d dimensions as

$$v_i = \frac{1}{n_i} \sum_{j \in C_i} x_j, \quad 1 \leq i \leq k, \quad (2)$$

where C_i is the ith cluster comprising of n_i entities. Intracluster similarity can be quantified as

$$S_i = \frac{1}{n_i} \sum_{j \in C_i} \left\| v_i - x_j \right\|. \quad (3)$$

The distance between two clusters C_p and C_q can be measured in terms of the distance between their centroids as

$$D_{pq} = \left\| v_p - v_q \right\|. \quad (4)$$

Davies-Bouldin (DB) index in the feature space is defined as

$$J_2 = \frac{1}{c} \sum_{p=1}^{c} \left[\max_{\substack{1 \leq q \leq c \\ p \neq q}} \left\{ \frac{S_p + S_q}{D_{pq}} \right\} \right], \quad (5)$$

which takes values in the range 0 to 1; the smaller the value, the better the partition.

The aim of the relational clustering using Sammon mapping is to simultaneously map the relational data into an equivalent lower-dimensional object space and optimize the mapped data into clusters. For the sake of brevity this algorithm will be referred to as SMC-R, for Sammon mapping and clustering of relational data. In all of the experiments described here, the lower dimension is $q = 1$. Each mapped object y_j is represented by a binary string of length p, that is, it can lie between 0 and $2^p - 1$ and p can be chosen such that $2^p \gg n$, where n is the size of the dataset. A modified cluster label representation is used where each mapped object is appended by its cluster label. Again, in the experiments described here, it was assumed that clusters are fixed and less than 8 in any given dataset. Therefore, three bits can be used to represent the cluster label of the mapped object. A partition can therefore be represented by a binary string of size $n \times (p + 3)$. For large values of n, this representation is not computationally attractive, and therefore the clustering methodology described here will be implemented only on small to medium datasets ($n < 1000$). The binary bits representing the cluster labels use a restricted growth function (RGF) scheme. The restricted growth function (RGF) is a remedy for the degeneracy problem inherent in the cluster labels [13]. Consider a two-cluster dataset of five entities where entity 1, 3, and 4 belong to one cluster, and entities labeled 2 and 5 belong to the second cluster. A cluster-label representation of a partition could be {12112}. However, {21221} is the same partition. In other words, there is a many-to-one mapping of representations in the phenotype (partition) space. The RGF scheme reorders the labels such that entity 1 is always assigned to the cluster labeled 1, entity 2 is assigned either to the cluster labeled 1 or 2, but not 3 or later, and so on. For n entities grouped into c clusters, the cluster-label chromosome is a function $f : [n] \rightarrow [n]$ such that

$$\begin{gathered} f(1) = 1, \\ f(i+1) \leq \max\{f(1), \ldots, f(i)\} + 1, \quad 1 \leq i \leq n, \\ \max\{f(1), \ldots, f(n)\} = c. \end{gathered} \quad (6)$$

A particular chromosome can be decoded as follows: first p bits provide object information about the first entity on a 1-D line in the range $[0, 2^p - 1]$, the next three bits decode to their cluster label, and so on. After recombination, an additional step which reorders the cluster label according to the RGF scheme is required. Single-point crossover and random bit-flip mutation with predefined probabilities are used as genetic recombination operators. If, after reordering, an individual in the offspring population decodes to a partition with less than c clusters, clusters are split up in two (one random cluster at a time) until a c partition is obtained. If, on the other hand, an individual represents a partition with more than c clusters, two clusters are picked at random and merged until a c partition is obtained. This step is combined with the RGF reordering step and is implemented after the entire offspring population is generated. The algorithm is described as follows.

(1) A random population of P individuals is initialized; each individual is a binary string of size $n \times (p + 3)$. Parameters such as number of generations G, probabilities of mutation, and crossover p_m and p_c, respectively, are fixed. Generation counter is initialized to 0.

(2) A mating pool is created using a tournament selector operator of size 2; two individuals are picked at random from the parent population, and the one with the higher fitness is inserted into the mating pool. The fitness of an individual is defined as

$$F_1 = \frac{\alpha}{J_1} + \frac{\beta}{J_2}, \tag{7}$$

where α and β are constants that weight the relative importance of Sammon mapping accuracy to cluster assignment accuracy, and J_1 and J_2 are given in (1) and (5), respectively. The size of the mating pool is the same as the population size.

(3) A pair of individuals is then picked sequentially from the mating pool, and two offspring individuals are created using recombination operators with probabilities p_c and p_m.

(4) The offspring population is merged with the parent population, and the combined population is ranked according to individual fitness. The top half of the population is retained as the new generation.

(5) Generation counter is incremented, and steps (2)–(5) are repeated till generation counter = G.

5. Density-Based Partitional Clustering

The attractiveness of DBSCAN lies in its simplistic framework and its robustness to outliers; however, the algorithmic implementation has issues which this paper deals with. This is conceptually similar to partitional clustering using the minimum sum of squares error criterion or the least squares (LSs) measure which states that the best partition is the one which minimizes intracluster distances although the criterion does not lend itself very well to implementation. Approximate iterative schemes such as k-means have been developed to deal with implementation of the LS measure. DBSCAN divides the entities to be clustered into three groups—core entities, border entities, and noise, based on a measure of how centrally (or noncentrally) located the points are in dense regions. A few definitions are in the following order.

Definition 1 (ε-neighborhood of an entity). The ε-neighborhood of an object x defined by $\varepsilon(x)$ is a collection of entities whose distance from x is no more than a predefined threshold ε, which is one of the two user-defined parameters.

Definition 2 (core entities). An object x_a is a core object if the size of the ε-neighborhood of x_a is larger than a predefined threshold *MinPts*, which is the second of the two user-defined parameters.

Definition 3 (directly density-reachable and border entities). An object x_b is directly density-reachable from a core object x_a if x_b belongs to the ε-neighborhood of x_a, that is, $x_b \in \varepsilon(x_a)$. If x_b itself is not a core object (according to Definition 2), then such directly density-reachable objects are termed border objects.

Definition 4 (density-reachable). An object x_c is density-reachable from another object x_b if there exists a chain of objects starting from x_b to x_c such that each object in the chain is directly density-reachable from the previous object in the chain.

Definition 5 (density-connected). An object x_c is density-connected to an object x_b if there exists another object x_d which is density-reachable from both x_b and x_c.

Definition 6 (cluster). A cluster is defined as soon as a core object is located. The two requirements to expand a cluster are that all directly density-reachable objects be part of that cluster and all objects in a cluster be at least density-connected to each other. If two core objects are in each other's ε-neighborhood, they belong to the same cluster.

Definition 7 (noise). Noise objects are those that are neither core nor border objects and do not belong to any cluster.

As can be seen from the definitions, reasonable estimates for ε and *MinPts* are critical to the success of DBSCAN. When different regions in the data have different densities, the same value of ε and *MinPts* may not always be the prudent choice. Modifications of DBSCAN include variants such as OPTICS [14] and DBCLASD [15]. The former uses additional definitions of core distance and reachability distance to cluster spatial databases which have regions of differing densities, while the latter assumes that entities inside a cluster follow a uniform distribution which reduces the dependence on user-specified parameters. In the next section, a new algorithm called DBSCAN-RC (relational clustering) is proposed which evolves the original DBSCAN estimates of ε and *MinPts*.

The two user-defined parameters in DBSCAN, ε and *MinPts,* are evolved using a genetic algorithm, and the resulting algorithm is simply called DBSCAN-RC. The original DBSCAN is recast as a relational clustering problem here. The relational data is modified as

$$R_{jk} = 1 - \left(\frac{r_{jk}}{r_{\max}}\right) \quad r_{\max} = \max_{1 \le j,k \le n} \left(r_{jk}\right). \tag{8}$$

A multiparameter, mapped, fixed point coding scheme is used for representing potential solutions to the clustering problem. The first parameter encodes the value of ε, and the second parameter encodes for *MinPts*, with the result that all encodings are the same length. In this implementation presented later, both parameters use eight each.

In the relational space, the neighborhood parameter ε is defined as a proximity parameter ranging between 0 and 1—a value of 0 implying that no entity is a core entity, a value of 1 meaning that all entities are core entities, and for any other ε, an entity o_j is a core entity with respect to *MinPts* if

$$\sum_{\substack{k=1 \\ k \ne j}}^{n} Z_k > MinPts, \quad Z_k = \begin{cases} 1, & \text{if } R_{jk} \ge \varepsilon, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

MinPts is the minimum limit on the number of entities that must be within the ε distance for an entity to be classified

as a core entity. It can theoretically range from 0 to n—a value of 0 means that any entity is a core entity as long as it has at least one other entity proximal to it above the ε-proximity threshold, and a value of n means that unless the ε-proximity threshold is unity, no entity is ever a core entity. For any other value in the range 0 to n, the encoded parameter is redefined as $(1 - MinPts/n)$. This redefinition scales the second parameter such that it always lies between 0 and 1. For example, with $n = 200$, if the first parameter in an individual (partition) in the population decodes to 0.35 and the second parameter to 0.45 (i.e., $MinPts = 110$), an entity will only be classified as a core entity only if there are at least 110 other entities with proximities of more than 0.35 with respect to it.

In the absence of cluster prototypes in the relational space, cluster assignment functionals can be used as measures of the quality of partition. The relational fuzzy c-means minimization functional is given by,

$$J^{fcm} = \sum_{i=1}^{c} \frac{\sum_{j=1}^{n} \sum_{k=1}^{n} u_{ij}^{m} u_{ik}^{m} r_{jk}^{2}}{\sum_{j=1}^{n} u_{ij}^{m}}, \tag{10}$$

where c is the number of clusters, $U = \{u_{ij};\ 1 \le i \le c,\ 1 \le j \le n\}$ is a membership function that quantifies the degree of belongingness of entity o_j in clusters C_i, and m is a parameter called the fuzzifier. Individual memberships assume values between 0 and 1. The membership function is constrained such that sum total of the membership of an entity across all clusters is unity, and the sum total of memberships of all entities in any cluster is more than zero (this ensures that there are no empty clusters). The previous equation is a general case, of which the crisp (nonfuzzy) case is a particular one—each entity is part of only one cluster (with a membership of one) and has zero membership in all other clusters. The relational hard c-means minimization functional per cluster is

$$J_3 = \frac{1}{c} \sum_{i=1}^{c} \frac{\sum_{j \in C_i}^{n} \sum_{k \in C_i}^{n} R_{jk}^{2}}{n_i}. \tag{11}$$

For fixed values of c, a small value of the above functional denotes a good partition. The nature of scaling r_{jk} to R_{jk} and the fact that the functional value is divided by the total number of cluster ensure that J_3 is always between 0 and 1.

In addition, a penalty is imposed on a partition that classifies an unusually large number of entities as noise. This requires an assumption about the possible contamination (noise) in the data—one reasonable assumption is that at least half the data will form meaningful clusters; a partition is not considered unless the size of the clustered data is more than $0.5n$. However, in experiments reported in this paper, a linear weighted function of the inverse of J_3 and the number of non-noise entities considered by the partition are used as a fitness function

$$F_2 = \frac{\delta}{J_3} + \frac{\eta}{n} \left(\sum_{i=1}^{k} n_i \right), \tag{12}$$

where δ and η are weights such that $\delta + \eta = 1$. While the first term rewards partitions with smaller J_3 values, the second term rewards those that classify more entities as either core or border entities (less noise) on the assumption that a vast majority of entities are part of good clusters. More discussion follows in the next section. The clustering algorithm is the same as the one proposed in Section 4, the fitness function given by (12) replacing the one in (7).

Table 1: Implementation parameters for clean three-cluster data.

	SMC-R	DBSCAN-RC
Length of the chromosome	9000	16
Population size, P	100, 200	100, 200
Number of generations, G	30	30
User-defined fitness parameters	$\alpha = \beta = 0.5$	$\delta = 1, \eta = 0$
Probability of crossover, p_c	0.95	0.98
Probability of mutation, p_m	0.05	0.005

6. Computational Results

The basic framework of the two algorithms presented here is the same—the differences lie in the representation used and the way in which the fitness function is calculated. In addition, SMC-R is noise-sensitive and is designed specifically for smaller datasets in higher dimensions; while DBSCAN-RC is robust against noise, and the sequential nature of DBSCAN has been shown to scale very well for large datasets. In this section, results are presented for two three-cluster synthetic datasets, one noisy and one free of noise. Both datasets are generated from Gaussian functions in eight dimensions. This is followed by providing DBSCAN-RC clustering results for two moderately large datasets from a literature.

The fitness functions in (7) and (12) require the user to predefine weights (α and β in SMC-R and δ and η in DBSCAN-RC) which are not trivial either. In this paper, values that work after limited experimentation were chosen and will be described later in this section.

6.1. Synthetic Three-Cluster Data. Three Gaussian clusters each comprised of 300 eight-dimensional vectors are generated. The three clusters are centered around 1, 4, and 7, respectively, in all eight dimensions. The data has intra-dimensional variance of 0.5 and interdimensional variance of 0.05. Each cluster contributes 300 objects to the dataset for a total of 900 objects. The object labels are randomized after generation. The object data so generated was then converted to relational (proximity) data using Euclidean distance measure. Clustering is performed using SMC-R and DBSCAN-RC with two different population sizes ($P = 100$, 200) and total number of generations $G = 30$. Parameters of the algorithms are shown in Table 1. Simulations were run on the 64-bit PyScripter IDE on a dual core 3.16 GHz processor with 8150 MB RAM. Results are presented in Tables 2 and 3 for two specific combinations of P and G. Ten runs were conducted with identical initial populations in both cases. While data about optimal partition found (yes or no) and number of generations till optimal partition was first uncovered is presented only for the best among the 10 runs,

Table 2: Clustering results for clean three-cluster data ($P = 100$, $G = 30$).

	SMC-R	DBSCAN-RC
Optimal partition found (best case)	Yes	No
Number of generations till optimal partition found (best case)	27	—
Ratio of average fitness of terminating generation to fitness of random initial generation (average)	15.2	13.1

Table 3: Clustering results for clean three-cluster data ($P = 200$, $G = 30$).

	SMC-R	DBSCAN-RC
Optimal partition found (best case)	Yes	Yes
Number of generations till optimal partition found (best case)	18	25
Ratio of average fitness of terminating generation to fitness of random initial generation (average)	18.7	14.5

Table 4: Implementation parameters for noisy three-cluster data.

	SMC-R	DBSCAN-RC
Length of the chromosome	10000	16
Population size, P	50–200	50–200
Number of generations, G	10, 20, 30, 50	10, 20, 30, 50
User-defined fitness parameters	$\alpha = \beta = 0.5$	$\delta = 0.5$, $\eta = 0.5$
Probability of crossover p_c	0.95	0.98
Probability of mutation p_m	0.05	0.005

the ratio of average fitness of the terminating generation to the initial generation is averaged over 10 runs.

The dataset was then corrupted by 100 uniformly distributed noise vectors in the range (1, 7) in all eight dimensions. In the eight-dimensional feature space, the Gaussian clusters are comprised of entities that are in dense regions, and the uniformly distributed noisy entities are in regions that are less dense. The ratio of the mean fitness of the terminating population to the initial population is measured and averaged over 10 random runs from the same initial population. The population size was varied from 50 to 200 in increments of 10 to track the effect of population size. Other parameters are listed in Table 4. The variation of the average fitness ratio with population size for four different values of maximum generations is shown in Figure 1.

No significant gains are achieved as the maximum number of generations is increased from $G = 30$ to $G = 50$. Likewise, $P = 100$ is a significantly better choice than $P = 90$ for both $G = 30$ and $G = 50$; however, the average fitness ratio does not markedly improve beyond $P = 100$. The choices are not generalizable, but the plots provide an empirical basis to judiciously select P and G for other similar datasets. In all runs except for $P < 100$ for $G = 10$, DBSCAN-RC identifies the optimal partition, defined as the one that identifies the three Gaussian clusters and the 100 uniformly distributed noise entities (within a 2% misclassification error). SMC-R does not correctly identify the optimal partition in any of the runs. The results show that while SMC-R is more efficient for smaller, noise-free data than DBSCAN-RC (possibly faster too, although CPU usage time is not reported), the opposite is true for clustering with noisy data as expected. The results

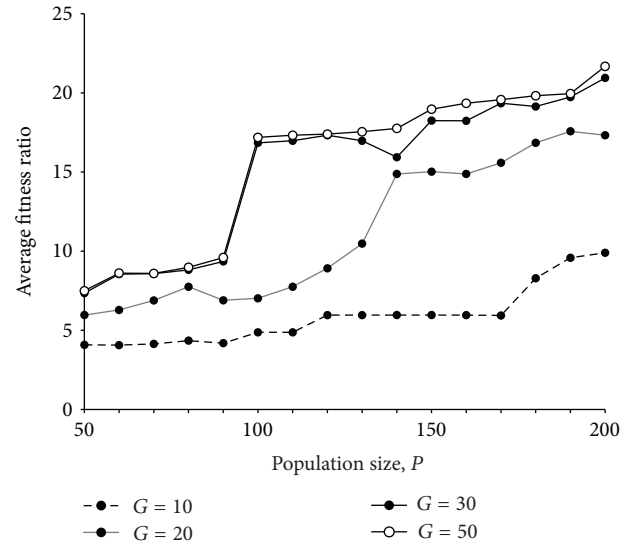

Figure 1: Ratio of the average fitness of the terminating population (at generation G) to the average fitness of the initial population plotted as a function of the population size P.

provide a justification for the two algorithms and verification of the proof of concept.

6.2. Wisconsin Breast Cancer Data. The Wisconsin breast cancer database is available at the UCI Machine Learning Repository [16] and has been analyzed as a two-class problem [17] and as a multiobjective optimization problem with unknown number of clusters in [8]. Each entity is defined by nine attributes (ranks in the range of 1–10) and an associated class label which are ignored during clustering. There are 699 total cases out of which 16 have a single missing feature. These missing features are randomly given "unusually large" values, these 16 cases can therefore be treated as noisy entities. Hamming distance was considered as the distance measure because the attributes take rank values. The maximum hamming distance between two entities is 81 when all nine of their attributes take maximal ranging values; the minimum distance being zero. The problem was analyzed using a population size of 300 running for 300 generations

with binary-bit representations, and the initial population was randomly generated using biased generators [8]. In this paper, two simulations are run using $P = 100$ and $P = 500$. The algorithm is stopped when the average fitness of the top half of the individuals converged within a predefined threshold (10^{-4} in here).

The simulations with $P = 100$ converged within 30–50 generations with near-optimal solutions appearing as early as 25th generation in the best case. In this case, near-optimal solutions are defined as partitions that correctly classify the noisy entities and have the small percentage misclassification in the nonnoisy part of the data. Due to the nature of the data, 95-96% classification accuracy has been reported in the literature, which was also found to be the maximum limit of accuracy in all the simulations using DBSCAN-RC. The simulations with $P = 500$ not only converged quicker than those with $P = 100$ (as is to be expected), but the populations also lost diversity considerably quickly. Although most of the individuals by the 25th generation encoded for two clusters and were near-optimal, almost a fourth (125–140 individuals) of the population were also near-optimal but encoded for five clusters. Similar results have been reported in the literature [18, 19]. However, this smaller subset did not constantly appear in simulations with $P = 100$. For almost similar results (and trends), the present implementation involves much less computational effort than previous studies.

6.3. Iris Data Set with Simulated Noise. The Iris dataset has been a benchmark set for machine learning tasks. The same set of four Iris datasets—the original and three contaminated datasets from [8, 9] are revisited here. The original Iris dataset consists of 150 samples of flower from the Iris species Setosa, Versicolor, and Virginica; 50 samples from each species are categorized according to four features—sepal length, sepal width, petal length, and petal width. Setosa is linearly separable from the others, but Versicolor and Virginica are not, which very often results in clustering algorithms uncovering two clusters in the Iris dataset instead of three. The Iris dataset and its variants were partitioned using a substantially large population evolving over quite a number of generations in [8, 9]. There was also the added issue of long binary bit representations used in previous studies; the constant length two-parameter representation used here is computationally a lot more attractive.

In this implementation of DBSCAN-RC, Euclidean distance is used as the basis of creating the proximity matrix. The four features are first scaled before being used in the distance metric. For $P = 50$, the results of DBSCAN-RC are compared to the results using the genetic algorithm-based clustering method that used a two-part chromosome and a two-tier fitness evaluation [9], abbreviated as 2GA for convenience and presented in Table 5. The second-tier fitness function used in 2GA is the fuzzy silhouette width which is similar to DB index in that it is also a measure of intracluster compactness and intercluster distances. The convergence criterion in 2GA is when the average second-tier fitness of a mating pool converges; a mating pool consists of top 12 individuals in a population of size 50. DBSCAN-RC terminates when the average fitness of the top 12 individuals in the best population at any generation converges below a threshold of 10^{-4}. The final population of best individuals is decoded, and phenotypes (partitions) are evaluated. The partitions are seen to correspond either to two or three clusters, with percentages reported in Table 5. Also reported are specificity and sensitivity values. Specificity or the true negative rate is the measure of identifying artificially added noise correctly (if all noise entities have been identified, the specificity is 100), and sensitivity or the true positive rate is the measure of identifying true data correctly (sensitivity is 100 if all the real Iris data is identified as core or border entities).

Unlike the two previous studies, the discrimination power of DBSCAN-RC does not deteriorate rapidly as contamination increases. In fact, the specificity and sensitivity of both classes (solutions encoding for $c = 2$ and for $c = 3$) are considerably better than those obtained with 2GA and reported in [9]. The fittest individual in the converged best population encodes for $c = 2$ more often than $c = 3$, a fact reported in [9] as well.

Table 5: Clustering results for contaminated Iris datasets.

Iris dataset with $n = 165$, 10% noise		
	2GA	DBSCAN-RC
Generations till convergence, G	51	32
% individuals with $c = 2$	16	10
% individuals with $c = 3$	84	90
Average specificity (family of $c = 2$)	80.00	93.33
Average specificity (family of $c = 3$)	86.67	93.33
Average sensitivity (family of $c = 2$)	88.00	91.33
Average sensitivity (family of $c = 3$)	94.67	97.33
Iris dataset with $n = 180$, 20% noise		
	2GA	DBSCAN-RC
Generations till convergence, G	57	36
% individuals with $c = 2$	25	10
% individuals with $c = 3$	67	88
Average specificity (family of $c = 2$)	83.33	90.00
Average specificity (family of $c = 3$)	90.00	93.33
Average sensitivity (family of $c = 2$)	86.67	90.66
Average sensitivity (family of $c = 3$)	93.33	96.66
Iris dataset with $n = 195$, 30% noise		
	2GA	DBSCAN-RC
Generations till convergence, G	62	37
% individuals with $c = 2$	33	12
% individuals with $c = 3$	67	88
Average specificity (family of $c = 2$)	73.33	91.11
Average specificity (family of $c = 3$)	77.77	95.55
Average sensitivity (family of $c = 2$)	85.33	90.66
Average sensitivity (family of $c = 3$)	88.00	92.66

7. Discussion

The primary contribution of this paper is presenting the framework for two relational clustering methodologies using evolutionary algorithms. A simplified constant length representation of a partition is also proposed that could potentially encode for a wide range of c values. Genetic representations used for clustering problems, where the number of clusters is unknown, have been of variable length. The constant length chromosome is made possible by including two density-related parameters instead of encoding for prototype location or cluster labels. The density-based clustering method, DBSCAN is also robust against outliers in the data, which makes it suitable for real world applications where data are almost always contaminated. Moreover, DBSCAN and its variants have been originally defined in feature (vector or entity) space; in this paper, a new relational version of the algorithm is presented.

References

[1] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD '96)*, AAAI Press, New York, NY, USA, 1996.

[2] R. J. Hathaway and J. C. Bezdek, "Nerf c-means: non-Euclidean relational fuzzy clustering," *Pattern Recognition*, vol. 27, no. 3, pp. 429–437, 1994.

[3] R. J. Hathaway, J. W. Davenport, and J. C. Bezdek, "Relational duals of the c-means clustering algorithms," *Pattern Recognition*, vol. 22, no. 2, pp. 205–212, 1989.

[4] R. N. Davé and S. Sen, "Robust fuzzy clustering of relational data," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 6, pp. 713–727, 2002.

[5] E. Falkenauer, *Genetic Algorithms and Grouping Problems*, John Wiley & Sons, Chichester, UK, 1998.

[6] E. R. Hruschka, R. J. G. B. Campello, A. A. Freitas, and A. C. P. L. F. de Carvalho, "A survey of evolutionary algorithms for clustering," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 39, no. 2, pp. 133–155, 2009.

[7] O. Nasraoui and R. Krishnapuram, "Clustering using a genetic fuzzy least median of squares algorithm," in *Proceedings of the 16th Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS '97)*, pp. 217–221, September 1997.

[8] A. Banerjee, "Robust fuzzy clustering as a multi-objective optimization procedure," in *Proceedings of the 28th Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS '09)*, June 2009.

[9] A. Banerjee, "An improved genetic algorithm for robust fuzzy clustering with unknown number of clusters," in *Proceedings of the 29th Annual North American Fuzzy Information Processing Society Conference (NAFIPS '10)*, July 2010.

[10] N. R. Pal, V. K. Eluri, and G. K. Mandal, "Fuzzy logic approaches to structure preserving dimensionality reduction," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 3, pp. 277–286, 2002.

[11] T. A. Runkler, "Fuzzy nonlinear projection," in *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE '03)*, pp. 863–868, May 2003.

[12] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 1, no. 2, pp. 224–227, 1978.

[13] A. Tucker, J. Crampton, and S. Swift, "RGFGA: an efficient representation and crossover for grouping genetic algorithms," *Evolutionary Computation*, vol. 13, no. 4, pp. 477–499, 2005.

[14] M. Ankerst, M. M. Breunig, H. P. Kriegel, and J. Sander, "OPTICS: ordering points to identify the clustering structure," in *Proceedings of the SIGMOD International Conference on Management of Data*, pp. 19–40, 1999.

[15] X. Xu, M. Ester, H. P. Kriegel, and J. Sander, "Distribution-based clustering algorithm for mining in large spatial databases," in *Proceedings of the 14th International Conference on Data Engineering*, pp. 324–331, February 1998.

[16] C. J. Merz and P. M. Murphy, "UCI Repository of Machine Learning Databases," University of California, Irvine, Calif, USA, 2007, http://www.ics.uci.edu/~mlearn.

[17] E. R. Hruschka and N. F. Ebecken, "A genetic algorithm for cluster analysis," *Intelligent Data Analysis*, vol. 7, pp. 15–25, 2003.

[18] A. Banerjee and S. J. Louis, "A genetic algorithm implementation of the fuzzy least trimmed squares clustering," in *Proceedings of the IEEE International Conference on Fuzzy Systems (FUZZ-IEEE '07)*, July 2007.

[19] R. Kothari and D. Pitts, "On finding the number of clusters," *Pattern Recognition Letters*, vol. 20, no. 4, pp. 405–416, 1999.