A Recursive Clustering Methodology using a Genetic Algorithm

Amit Banerjee and Sushil J. Louis, *Member, IEEE*

***Abstract*—This paper presents a recursive clustering scheme that uses a genetic algorithm-based search in a dichotomous partition space. The proposed algorithm makes no assumption on the number of clusters present in the dataset; instead it recursively uncovers subsets in the data until all isolated and separated regions have been classified as clusters. A test of spatial randomness serves as a termination criteria for the recursive process. Within each recursive step, a genetic algorithm searches the partition space for an optimal dichotomy of the dataset. A simple binary representation is used for the genetic algorithm, along with classical selection, crossover and mutation operators. Results of clustering on test cases, ranging from simple datasets in 2-D to large multidimensional datasets compare favorably with state of the art approaches in genetic algorithm-driven clustering.**

I. Introduction

THE organization of data into groups is one of the most fundamental modes of understanding and learning [1]. Cluster analysis is the formal study of mechanisms, algorithms and methods for classifying entities. Unlike discriminant analysis or pattern recognition, cluster analysis does not use category labels; instead the objective is to achieve a valid organization of the data.

In broad terms, cluster analysis is composed of three tasks, viz. clustering tendency, clustering and cluster validation. Clustering tendency refers to the problem of deciding whether data exhibit a predisposition to cluster in natural groups without actually identifying the groups. In the absence of a structure, it is wasteful to seek clusters, more so because clustering algorithms divide the data into clusters even in the absence of natural groups. Clustering tendency tests are usually performed before the actual process of clustering itself. Clustering algorithms generate an efficient representation of the data by classifying them into subsets that have meaning in the context of a particular problem. A proximity measure that objectifies the relationship between pairs of entities in the data is essential for clustering. Based on the way a structure is enforced on (or uncovered in) the data, clustering algorithms can be hierarchical or partitional. Both techniques produce a finite partition of the data into a desired (often fixed) number of clusters. Partitions are then evaluated by one or more validation mechanisms that provide objective information about the *goodness* of the partition. Since the actual number or the nature of clusters in a data set is hardly ever known *a priori*, clustering algorithms are implemented on a range of values corresponding to number of clusters, followed by cluster validation studies on every partition. The validation measures usually indicate if a particular partition is superior to others.

In this paper, we present a methodology that seamlessly puts the three disjoint stages of cluster analysis into one workable framework with a genetic algorithm-driven search in the partition space. We approach the problem of clustering from a purely assignment point of view; the problem definition is – assign n data points in d-dimensions to k clusters, so that a certain underlying notion of homogeneity within a cluster and heterogeneity across clusters is maximized. The assignment of n points to k clusters is purely a combinatorial one [2], given by

$$S(n,k)=\frac{1}{k!}\sum_{j=0}^{k}(-1)^{j}\binom{k}{j}(k-j)^{n}. \qquad (1)$$

This results in approximately 2.5×10^{15} possible solutions for a simple problem of assigning $n = 25$ entities to $k = 5$ clusters. When the number of clusters for assignment is not known *a priori*, there exists more than 4.6×10^{18} different ways to assign 25 entities to clusters (ranging from a one-cluster solution to all-singleton solutions). This problem is NP-complete and attempting to find a global optimum is usually not computationally viable even for moderately sized datasets. Genetic algorithms (GA) are considered effective for NP-complete search and optimization problems and guarantee at least locally optimal solutions in reasonable time [3].

The first reference to the use of evolutionary techniques in clustering can be found in [4], where an adaptive strategy evolves populations of non-overlapping classifications using selection, inversion and mutation. A K-means clustering is performed taking the membership degrees and prototype locations as the parameters for the GA in [5]. (Prototype locations in K-means and related methodologies are usually cluster centers specified by a mean location in d-dimensions.) In an extension [6], the prototype locations are encoded as binary strings and genetic operators then operate to optimize location instead of operating on the membership

Manuscript received March 15, 2007. This material is based upon work supported by the National Science Foundation under grant no. 0447416.

Amit Banerjee is with the Evolutionary Computing Systems Laboratory, Department of Computer Science and Engineering, MS/171, University of Nevada Reno, NV 89557 USA (e-mail: banerjee@ cse.unr.edu).

Sushil J. Louis is with the Evolutionary Computing Systems Laboratory, Department of Computer Science and Engineering, MS/171, University of Nevada Reno, NV 89557 USA (phone: +1-775-784-4315, fax: +1-775-784-1877, e-mail: sushil@cse.unr.edu).

matrices. The prototype location is also encoded in [7] and [8] as real-numbered ordered pairs of length $k \times d$, where k is the number of clusters and d is the dimensionality of the dataset. An elitist model of the GA was used in [9] to create decision boundaries (approximated by piecewise linear segments) to classify data. These decision boundaries are encoded as strings and the optimization is driven by a function that minimizes misclassification. They later developed the GA-classifier that used a variable length string to automatically identify the minimum number of classifying hyperplanes in the data [10]. Murthy and Chowdhury [11] use an n-bit cluster label-based string, where n is the number of data points to be clustered. Consider a simple case where $n = 8$ data points are clustered into $k = 3$ clusters such that data points x_1, x_3, x_6 form cluster-1, x_2, x_4 form cluster-2 and x_5, x_7, x_8 form cluster-3. The chromosome representing this case is 12123133 – the alleles are cluster labels. It is however, not clear as to the nature of the fitness function used to evaluate the chromosomes in [11]. A similar representation is used in [12] and [13]. A variance-ratio based goodness-of-fit criterion is maximized in [12], which objectifies external cluster isolation and internal cluster homogeneity. A variant of the Average Silhouette Width [14] is used as a maximization function in [13]. Additional constraints have to be imposed on crossover and mutation operations in order to produce meaningful and valid offspring when such a cluster label-based representation is used. To work around this problem of expensive genetic operators, a K-means type gradient descent search operator replaces the crossover operator and the classical mutation operator is replaced by a biased mutation operator specific to clustering, called the distance-based mutation operator [15]. A binary representation called the Boolean Matching Code (BMC) was used in [16]. The same $n = 8$, $k = 3$ case presented above is coded as [10100100, 01010000, 00001011], and a specialized crossover operator called the Single Gene Crossover (SGC) operator is proposed. More recently, a genetically guided hierarchical cluster-merging operator was introduced to merge randomly fragmented subsets of the data into k-clusters [17]. An incremental hierarchical clustering algorithm is combined with a standard GA in [18]; however, the GA is used only to improve the quality of the output of the hierarchical clustering scheme. Evolutionary programming approaches to clustering have been investigated in [19]. A major shortcoming of the K-means clustering algorithm – its overt dependence on initialization, is addressed using a genetically algorithm in [20]. In [21], a GA was used to guide search for valid prototype locations in a noisy dataset using the least median of squares criteria for clustering.

Apart from the need for good initialization for gradient descent search techniques such as K-means, and the assumption that data is not corrupted with noise (and outliers), almost all clustering routines assume that the number of clusters to be found is known *a priori*. They work around this assumption by performing clustering over a range of k-values and then choosing the best partition. Many such studies assume fixed length binary coding, where the length of the string depends on the number of clusters e.g. [16], or parameterize the prototype locations instead of assignments or cluster labels [7], [8]. On the other hand, studies involving uncovering structure in the data without the explicit assumption use complicated representations, mainly the cluster label-based notation with specialized crossover and mutation operators, with a series of checks and bounds to prevent invalid offspring. In this paper, we present a framework for cluster analysis using a GA with a binary representation and classical reproductive operators.

II. Cluster Analysis

The use of simple encoding systems causes problems of redundant codification and context insensitivity [22]. This has led researchers to devise complicated representations and specialized operators (and in many cases radically different GAs) for clustering problems. The cluster label-based n-bit encoding is simple compared to parameterization of prototype location or encoding linear decision boundaries. Apart from being longer (n-bits as opposed to $k \times d$ bits for encoding the prototype parameters), such a representation also means that many genotype translate to a unique phenotype; a simple example is shown below,

Genotype1: 11223344
Genotype2: 22113344
Genotype3: 11332244
Genotype4: 22334411

All the four genotypes above, represent the case where $\{x_1, x_2\}$, $\{x_3, x_4\}$, $\{x_5, x_6\}$ and $\{x_7, x_8\}$ form four clusters. It is irrelevant to the clustering algorithm if the cluster formed by $\{x_1, x_2\}$ is called cluster-2 instead of cluster-3 or cluster-1. In other words, the notion of cluster labels built into the representation makes little intuitive sense. Such representations have spawned off a set of pre-treatment methodologies to make the representations suitable for genetic operators, such as the consistent algorithm for reassignment [23]. It is also argued that traditional GAs (e.g. the canonical GA), when applied to clustering tasks, often perform just a random search for the best solution [13]. That however, is contingent on the objective function that evaluates string fitness relative to each other. An objective function that biases the search towards isolated packed clusters within a structured data can ensure that the GA does not search randomly.

The clustering problem is treated a simple assignment problem of assigning n data points to k clusters. The value of k is not known but the motivation is to uncover a partition that best represents the underlying structure. We propose a recursive scheme to uncover closely-packed isolated clusters

one at a time until all the clusters have been identified, resulting in a partition that closely represents the structure of the data. The recursive scheme uses a clustering tendency measure to terminate when all the clusters have been identified. Within each recursive level, a GA is used to dichotomize the dataset, until the divisions represent the actual clusters. In this section, we first present the clustering tendency technique used as the termination measure, the objective function used by the GA to evaluate genotypes, a brief description of the GA used followed by a description of the clustering algorithm.

A. Clustering Tendency

The term clustering tendency refers to the problem of deciding whether data exhibit a predisposition of cluster into natural groups [1]. The test for tendency is also a test for spatial randomness, and there exist testing methodologies that test for structure in data (without actually identifying the clusters themselves). Such clustering tendency tests include scan tests, quadrat analysis, second moment structure tests and sparse sampling tests [1]. Sparse sampling tests are based on sampling origins randomly identified in a sampling window. Several tests involving sampling origins have been proposed in literature, based on a multitude of test statistics such as the Hopkins statistic, the Holgate statistic, the T-square statistic, the Eberhardt statistic and the Cox-Lewis statistic [1]. These statistics have been compared but little is known about any one of them outperforming the others. The Hopkins statistic is easy to use and comprehend and has been shown to be as good as the Holgate statistic [24] and has been used as a measure in a test for cluster validity [25]. Given n data points in a d-dimensioned space, the sparse sampling procedure randomly picks m datum ($m << n$) and calculates distances to nearest datum for each of these randomly picked datum – let w_i be the set of nearest neighbor distances for $1 \le i \le m$. Separately, the procedure also randomly injects m sampling origins into the dataset, and locates datum in the set which are closest to these randomly injected sampling origins – let u_i be the set of sampling origin to nearest datum distances for $1 \le i \le m$.
The Hopkins statistic is defined as,

$$H = \frac{\sum_{i=1}^{m} u_i^d}{\sum_{i=1}^{m} u_i^d + \sum_{i=1}^{m} w_i^d}. \qquad (2)$$

When the dataset is random, implying a lack of underlying structure, the value of $H \approx 0.5$, and for structured datasets, the value of H is close to unity. In the proposed methodology, this test is performed every time a dataset is divided into two parts. Consider a simple three-cluster case and let the clusters be labeled A, B and C. When the dataset is subject to the test of spatial randomness, we would expect $H \approx 1$, because of the presence of three distinct aggregated clusters. This data is then subject to division by an assignment routine driven by a GA that searches for the best dichotomy in the data. Let the first level of division create two subsets x and y, where x is composed entirely of cluster A and y is a combination of B and C. When x is tested for spatial randomness, we would expect $H \approx 0.5$, while for y we would expect $H \approx 1$. In other words, x is identified as a self-contained isolated cluster and removed from our consideration, and y is divided further until B and C are discovered. In general the algorithm terminates when $H \approx 1$ on all truncated datasets.

B. Objective Function

The objective function evaluates the fitness of individual strings (the encoding of the strings is explained later). Almost all partition evaluation functions provide some kind of measure of inter-cluster isolation and/or intra-cluster homogeneity. For a good partition, there should be appreciable inter-cluster isolation (or heterogeneity among clusters) and intra-cluster homogeneity. The homogeneity within a cluster is however, calculated by a minimizing function – usually the sum of distances between all the pairs of entities within a cluster. In fact, the popular K-means algorithm minimizes the sum of root mean squared distances of objects in a cluster from the cluster center. We use an objective function based on the Average Silhouette Width [14] and also used in [13]. The approach defines a *silhouette width* for each entity in the dataset, an average silhouette width for each cluster and overall average silhouette width for the total dataset. The silhouette width of a cluster is a measure of its tightness and separation. Similar measure of compactness (tightness) and isolation (separation) have also formed the basis of many cluster validation techniques e.g. [26]. The individual silhouette associated with an entity x_i in the dataset is given by,

$$S_i = \frac{b_i - a_i}{max(a_i, b_i)}, \qquad (3)$$

where a_i is the average dissimilarity of x_i to all other entities in the same cluster and b_i is the average dissimilarity of x_i to all entities in a cluster closest to its own. It also follows that $-1 \le S_i \le 1$. If $S_i \approx 1$, it means that x_i is assigned to a very appropriate cluster. If silhouette value is about zero, it means that that x_i could be assigned to another cluster. This implies that x_i lies equally far away from the two clusters (its own cluster as well as its closest neighboring cluster). If silhouette value is close to –1, it means that x_i is *misclassified* and should be assigned to another cluster. The overall average silhouette width for the entire dataset is defined as,

$$S = \sum_{i=1}^{n} S_i \ . \tag{4}$$

It can easily be seen that a high value of S (close to n) implies a good partition. There is also a unique value of S for every partition – of all the possible values of S the best partition is associated with the highest value. For a given partition, consider an entity x_i that belongs to cluster A. Let cluster A consist of m entities out of the possible n, including x_i. The average dissimilarity of x_i with all other entities of its own cluster (cluster A) is given by,

$$a_i = \frac{\sum_{k \in A} d_{ik}}{m-1} \tag{5}$$

where d_{ik} is the distance (or some measure of dissimilarity) between x_i and x_k. Consider another cluster B consisting of p entities. The average dissimilarity of x_i with all entities in B is given by,

$$b_{i,B} = \frac{\sum_{k \in B} d_{ik}}{p}, \tag{6}$$

And the average dissimilarity of x_i to all objects contained in the cluster closest to A is then given by,

$$b_i = min\left(b_{i,B}\right), B \neq A \tag{7}$$

When the entities in cluster A are similar, the value of a_i is small, implying homogeneity in A. Similarly, the higher the value of b_i, the higher the heterogeneity among x_i and entities in its neighboring cluster. If x_i belongs to a singleton (a cluster with only one entity), it is assumed that $S_i = 0$. Since genetic selection techniques require strings to have non-negative fitness values, we scale the range of S_i from [-1, 1] to [0, 2] by adding 1 to all S_i values before selection as,

$$S_i = \frac{b_i - a_i}{max\left(a_i, b_i\right)} + 1.0 \tag{8}$$

C. The Genetic Algorithm

We have used the classical GA [27] with stochastic tournament selection, simple one-point crossover and mutation operators. This is in contrast to very specialized GAs developed and used for clustering, including one that evaluates a complex population of schemas and individuals [28]. We also use the binary alphabet to encode for the dichotomy in the partition space. It has been shown that the binary alphabet provides the maximum number of schemata per bit of information of any coding technique [27]. The recursive uncovering of structure also allows for easy interpretation of the binary coding. At every truncation level, a population of binary strings is created randomly. Each individual string corresponds to a unique partition of the dataset at that level. At different truncation levels, the sizes of the datasets (truncated subsets) are different; hence the length of the binary string differs from one truncation level to the next. The strings are evaluated using (4) and (8), and given a non-negative fitness value, S ($0 \leq S \leq 2n$). Stochastic tournament selection, also called Wetzel ranking, is used to create a pool of highly fit strings. In the stochastic tournament selection, two strings are drawn from the population using the roulette wheel selection technique, evaluated against each other and the fitter string finds its way into the mating pool. In a related work [29] it has been shown that for cluster-assignment problems, stochastic tournament consistently outperforms roulette wheel selection and stochastic sampling methods of selection. Strings are then drawn at random from this mating pool for crossover and mutation. The newly created offspring completely replace the parent population and the process of selection, crossover, mutation and replacement is carried on for a fixed number of generations, after which the string with the maximum fitness is extracted from the population as the solution.

D. The Clustering Algorithm

The clustering methodology proposed here is essentially a search through the partition space. The search space is the set of all possible partitions of the dataset at any particular truncation level. A suitable threshold for the Hopkins statistic is chosen (usually between 0.75 and 0.85) and the parameters for the GA such as the population size, maximum number of generations, probability of crossover and mutation etc. are also fixed. The dataset is subject to the test of spatial randomness and if required, is divided into two by the clustering routine. The clustering is a simple assignation process driven by the GA on the evaluation function S. These truncated sets are then subject to the test of randomness (referred to by Hopkins(.) in the algorithm given below) individually. If the value of the Hopkins statistic is less than the pre-fixed threshold for the truncated datasets, they are further divided in two until the test of randomness proves that all truncated sets are random within themselves. The algorithm is given below,

Given dataset $\mathbf{X}_0 = \{x_i;\ 1 \leq i \leq n,\ x_i = \{x_{ij};\ 1 \leq j \leq d\}\}$;
Fix threshold statistic, H^T
Fix parameters for genetic algorithms, pop size = P, max gen = G;
Truncation level, $t = 0$;
$k = 1$, $l = 1$;
$x_l^t = \mathbf{X}_0$;
while ($l > 0$)
  $j = l$, $l = 0$;
  for ($i = 1$ to j)
    $l \rightarrow l + 1$;

```
H = Hopkins ( x_i^t );
if (H < H^T):
    N = size of ( x_i^t );
    {P_0} = Generate P random binary strings of N-bits;
        for (gen = 0 to G)
            Calculate Avg Silhouette Width for every string in {P_gen};
            Mating pool {M} = {};
            Select P strings from {P_gen} to form {M};
            Pick strings from {M} at random;
            Do crossover and mutation;
            gen → gen + 1;
            {P_gen} = P offspring;
        end for (gen)
        str = fittest string from {P_gen};
        x_l^{t+1} = { x_l^t ; str(h) = 1};
        l → l + 1;
        x_l^{t+1} = { x_l^t ; str(h) = 0};
    else:
        Y_k = x_i^t ;
        k → k + 1, l → l – 1
        i → i + 1
        t → t + 1
    end for (i)
end while (l > 0)
print (total clusters identified = k – 1);
for (i = 1 to k – 1)
    print cluster Y_i
    i → i + 1
end for (i)
```

III. SIMULATIONS

The proposed algorithm is implemented on clustering problems with three different datasets – the benchmark Ruspini data, the Iris plant data and the E. coli dataset. Even with the best of clustering algorithms, there is the possibility of occasional misclassifications – in our case partition misclassification by the GA-driven routine, might result in inconsistencies in results of the test of spatial randomness. The Hopkins statistic is fortunately found to be very robust to occasional misclassifications. On an average, if a random set of n entities is corrupted by a set of p entities ($< n/10$), resulting in an overall dataset that is not random (such is the result of a misclassification), the Hopkins statistic on such a dataset will indicate randomness if sampling size, $m < p$. In our experiments, we assume $m = n/15$ (rounded off to the nearest integer for $m > 1$; for $n < 15$, we assume $m = 1$). Note here that n is the size of the truncated dataset under consideration. Simulations with random datasets with less than 10% structured noise provides us with a threshold value, $H^T = 0.75$-0.85 to use in future experiments. We have used Euclidean distance as the measure of dissimilarity between entities x_i and x_k as,

$$d_{ik} = \sqrt{\sum_{j=1}^{d} (x_{ij} - x_{kj})^2} \quad . \tag{9}$$

A. Example 1 – Ruspini data

The Ruspini data is a benchmark dataset for classification and cluster analysis techniques. The dataset consists of 75 data points defined in 2-dimensions [14, p. 100] that divide into 4 clusters, as shown in fig 1. The best results were obtained for a population size, $P = 300$, $P_c = 0.95$, $P_m = 0.01$ when convergence was attained in 35 generations ($G = 100$) at the first level. The truncated sets identified are sets *A*-*D* and *B*-*C*. At the second level, the procedure converged in 15 generations on an average for the truncated set *A*-*D* and in 20 generations for the set *B*-*C*. We simulated 10 experiments with different random seeds to generate the initial population and found a narrower range of convergence (relative to generations) than reported in [13]. The convergence plot of the fitness function vs. generations (for the above mentioned parameter set) is shown in fig 2.

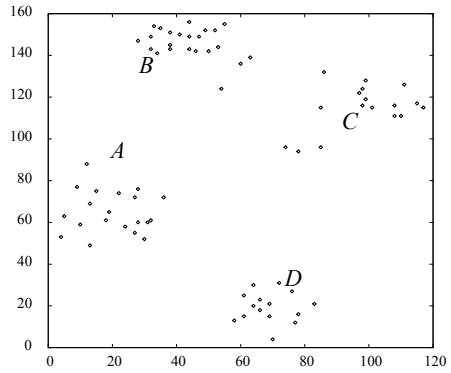

Fig 1. Ruspini data

B. Example 2 - Iris plant data

The Iris data consists of 150 random samples of flowers from the iris species *setosa*, *versicolor*, and *virginica*; for each species there are 50 observations for sepal length, sepal width, petal length, and petal width ($n = 150$, $d = 4$). This dataset was made famous by Fisher [30] in his work on the linear discriminant function. The class *setosa* is linearly separable from the others, but classes *versicolor* and *virginica* are not (as shown in fig. 3). Our recursive clustering procedure found two truncated sets after approximately 35 generations with different random initializations for $P = 600$, $P_c = 0.90$, $P_m = 0.01$ and $H^T = 0.85$. One truncated set corresponded to instances in the class *setosa* while the other set contained instances from *versicolor* and *virginica*. Although the test of spatial randomness indicated presence of further clusters in the latter ($H \approx 0.80$), the assignment routine driven by the GA fails to further divide the second set satisfactorily. In the best case, we obtain two clusters (corresponding to

versicolor and *virginica*) with almost 15% misclassification. Other researchers [13], [31], [32] have reported finding two instead of three clusters in the iris dataset. It has been inferred that many entities in the classes *virginica* and *versicolor* are very similar [13]. Using a binary representation, and simple reproductive operators, we have been able to perform as well as the Clustering Genetic Algorithm [13].

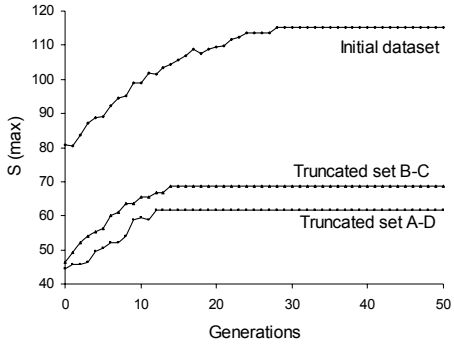

Fig. 2. Convergence Plot – Maximum Fitness vs. Generations. The first level of clustering was performed on the Initial dataset, resulting in two sets *A-D* and *B-C*. The second level of clustering on *A-D* and *B-C* uncovered the four clusters, *A*, *B*, *C* and *D*.

C. Example 3 – E. coli data

This dataset is available at the UCI Machine Learning Repository [33] and has been analyzed in [34]. The database is a collection of 336 instances of protein sequences in the E. coli bacteria defined by seven predictive attributes and a class label. The class label is the localization site of the protein in the E. coli. There are eight distinct localization sites (e.g. cytoplasm, inner membrane etc.) identified for the 336 proteins in the database. The truncated sets produced after the first level correspond to: (*i*) a class of proteins that localize in the cytoplasm and the periplasm (called the *cp-pp* subclass for clarity), and (*ii*) proteins that localize on the membrane (called the *im-om* subclass). The accuracy of classification is more than 93% at this stage (315 out of 336 protein sequences are correctly classified), with $P = 300$, $P_c = 0.95$, $P_m = 0.01$ and $H_T = 0.85$. Correcting for misclassifications at this stage, we classify the *cp-pp* class into two subsets – one corresponding to the 143 cytoplasm (*cp*) localization sites and the other 52 periplasm (*pp*) sites. The classification accuracy was almost 95% in this stage (with 11 out of 195 sequences misclassified). On the other *im-om* subclass, we identified a class corresponding to the 25 outer membrane localization sites (*omT*), and the other 116 sequences localizing at the inner membrane (*imT*), with almost 95% classification accuracy again. Further division of the *imT* subclass produced in the second level of truncation revealed the presence of two classes – one corresponding to the larger class of *im* sequences produced without signal sequence (referred simply by *im* [33]) and the smaller class consisting of the other inner membrane sequences (*imS*, *imL* and *imU* [33]). Subdivision of the *omT* subclass produces two clusters corresponding to *om* and *omL* [33]. The classification hierarchy is shown in fig. 4. With more than 90% overall accuracy, we have been able to identify six clusters in the E. coli data. This is in contrast to a classification accuracy of 81% reported in [34] that used an expert system classification algorithm with probabilistic reasoning.

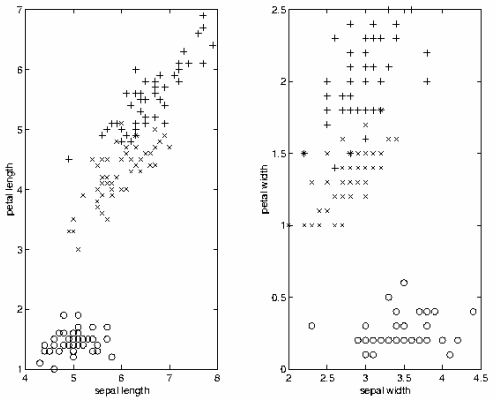

Fig 3. The Iris data plotted separately for length and width attributes for the 150 instances.

IV. Conclusions

We present a recursive clustering methodology driven by a genetic algorithm. The use of genetic algorithms for clustering goes back more than a decade; however, almost all GA-based clustering methodologies proposed in the literature attempt a single partition of the data, as opposed to uncovering the organization hierarchically. This has led to the development of highly specialized GAs and operators. In this paper, we have used the canonical GA in a recursive hierarchical methodology to uncover structure in data. We have also used a binary representation to encode the genotype – such a representation facilitates dichotomous classification in steps. We have also proposed using a test of spatial randomness as termination criteria and have illustrated our methodology with a simple statistic. The proposed methodology has also been compared with results of state of the art GA-based clustering, and has shown to be comparable. This given the fact that the methodology does not depend on specialized representations and operators, is very promising.

We are presently working on streamlining the proposed methodology that will combine the termination criteria and the fitness function into one evaluation function. We are also investigating robust versions of the proposed methodology that will allow recursive clustering of noisy datasets.

REFERENCES

[1] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice Hall, 1988.

[2] G. L. Liu, *Introduction to Combinatorial Mathematics*. New York, NY: McGraw-Hill College, 1968.

[3] Y. Park and M. Song, "A genetic algorithm for clustering problems," in *Proc. 3rd Annual Conf. Genetic Programming*, Morgan Kaufmann, 1998, pp. 668-575.

[4] V. V. Raghavan and K. Birchard, "A clustering strategy based on a formalism of the reproductive process in natural systems," in Proc. *2nd Annual Int. ACM SIGIR Conf. Information Storage and Retrieval*, Dallas, TX, 1979, pp. 10-22.

[5] J. C. Bezdek, S. Boggavarapu, L. O. Hall, and A. Bensaid, "Genetic algorithm guided clustering," in *Proc. 1st IEEE Conf. Evolutionary Computation*, Orlando, FL, 1994, pp. 34-39.

[6] J. C. Bezdek and R. J. Hathaway, "Optimization of fuzzy clustering using genetic algorithms," in *Proc. 1st IEEE Conf. Evolutionary Computation*, Orlando, FL, 1994, pp. 589-594.

[7] F. Klawonn and A. Keller, "Fuzzy clustering with evolutionary algorithms," *Int. J. Intell. Syst.*, vol. 13, pp. 975-991, 1998.

[8] U. Maulik and S. Bandyopadhyay, "Genetic algorithm-based clustering technique," *Pattern Recog.*, vol. 33, pp. 1455-1465, 2000.

[9] S. Bandyopadhyay, C. A. Murthy, and S. K. Pal, "Pattern classification with genetic algorithms," *Pattern Recog. Lett.*, vol. 16, pp. 801-808, 1995.

[10] S. Bandyopadhyay, C. A. Murthy, and S. K. Pal, "Pattern classification using genetic algorithms: Determination of H," *Pattern Recog. Lett.*, vol. 19, pp. 1171-1181, 1998.

[11] C. A. Murthy and N. Chowdhury, "In search of optimal clusters using genetic algorithms," *Pattern Recog. Lett.*, vol. 17, pp. 825-832, 1996.

[12] M. C. Cowgill, R. J. Harvey, and L. T. Watson, "A genetic algorithm approach to cluster analysis," *Computers and Mathematics with Appl.*, vol. 37(7), pp. 99-108, 1999.

[13] E. R. Hruschka and N. F. F. Ebecken, "A genetic algorithm for cluster analysis," *Intell. Data Anal.*, vol. 7, pp. 15-25, 2003.

[14] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley Series in Probability and Mathematical Statistics, New York, NY: John Wiley, 1990.

[15] K. Krishna and M. N. Murty, "Genetic K-means algorithm," *IEEE Trans. Syst, Man Cybern. B*, vol. 29(3), pp. 433-439, 1999.

[16] R. Cucchiara, "Genetic algorithms for clustering in machine vision," *Machine Vision and Appl.*, vol. 11, pp. 1-6, 1998.

[17] G. Garai and B. B. Chaudhuri, "A novel genetic algorithm for automatic clustering," *Pattern Recog. Lett.*, vol. 25, pp. 173-187, 2004.

[18] W. A. Greene, "Unsupervised hierarchical clustering via a genetic algorithm," in *Proc. IEEE Cong. Evol. Comput.*, Canberra, Australia, 2003, pp. 998-1005.

[19] M. Sarkar, B. Yegnanarayana, and D. Khemani, "A clustering algorithm using an evolutionary programming-based approach," *Pattern Recog. Lett.*, vol. 18, pp. 975-986, 1997.

[20] L. O. Hall, I. B. Ozyurt, and J. C. Bezdek, "Clustering with a genetically optimized approach," *IEEE Trans. Evolutionary Comput.*, vol. 3(2), pp. 103-112, 1999.

[21] O. Nasraoui and R. Krishnapuram, "A genetic algorithm for robust clustering based on a fuzzy least median of squares criterion," in *Proc. Annual Meeting of North American Fuzzy Inform. Process. Soc.*, Syracuse, NY, Sept. 1997, pp. 217-221.

[22] E. Falkenauer, *Genetic Algorithms and Grouping Problems*. New York, NY: John Wiley & Sons, 1998.

[23] E. R. Hruschka and N. F. F. Ebecken, "A clustering genetic algorithm for extracting rules from supervised neural network models in data mining tasks," in *Int. J. Computers, Systems and Signals, Special Issue: Knowledge Discovery from Structured and Unstructured Data*, vol. 1(1), A. P. Engelbrecht, Ed., Dec 2000, pp. 17-29.

[24] E. Panayirci and R. C. Dubes, "A test for multidimensional clustering tendency," *Pattern Recog.*, vol. 16(4), pp. 433-444, 1983.

[25] A. Banerjee and R. N. Dave, "Validating clusters using the Hopkins statistic," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Budapest, Hungary, July 2004, pp. 149-153.

[26] X. L. Xie and G. Beni, "A validity measure for fuzzy clustering," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-13(8), pp. 841-847, 1991.

[27] D. E. Goldberg, *Genetic Algorithms in Search, Optimizing and Machine Learning*. Boston, MA: Addison-Wesley, 1989.

[28] L. A. N. Lorena and J. C. Furtado, "Constructive genetic algorithm for clustering problems," *Evolutionary Comput.*, vol. 9(3), pp. 309-327, 2001.

[29] A. Banerjee and S. J. Louis, "A genetic algorithm implementation of the fuzzy least trimmed squares clustering,"in Proc. *IEEE Int. Conf. Fuzzy Syst.*, London, England, July 2007.

[30] R. A. Fisher, "The use of multiple measurement in taxonomic problems," *Annals of Eugenics*, vol. 7, pp. 179-188, 1936.

[31] R. M. Cole, "Clustering with genetic algorithms," M.S. thesis, Dept. Comp. Sc., Univ. Western Australia, 1998.

[32] R. Kothari and D. Pitts, "On finding the number of clusters," *Pattern Recog. Lett.*, vol. 20, pp. 405-416, 1999.

[33] C. J. Merz and P. M. Murphy, UCI Repository of Machine Learning Databases, Univ. California, Irvine, CA.
Available: http://www.ics.uci.edu/~mlearn

[34] P. Horton and K. Nakai, "A probabilistic classification system for predicting cellular localization sites of proteins," in *Proc. 4th Int. Conf. Intell. Syst. Mol. Biol.*, St. Louis, MO, 1996, pp. 109-115.

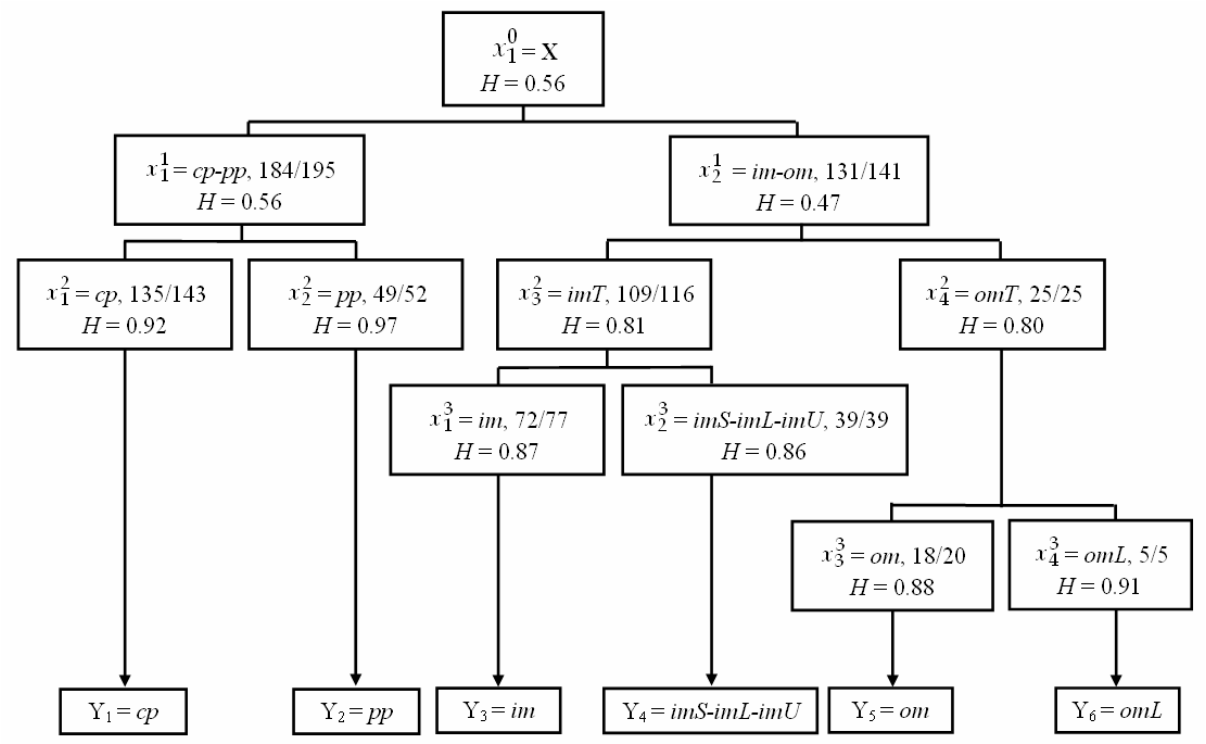

Fig. 4. Classification hierarchy for the E. coli data. There are three levels of truncation and six clusters identified.