A Bacterial Evolutionary Algorithm for Automatic Data Clustering

Swagatam Das[1], Archana Chowdhury[2] and Ajith Abraham[3,4]

[1]Department of Electronics and Telecommunication Engineering, Jadavpur University, Kolkata, India
swagatamdas19@yahoo.co.in
[2]M.P.C.C.E & T,Bhilai, Madhya Pradesh, India
archana81077@yahoo.com
[3]Center of Excellence for Quantifiable Quality of Service, Norwegian University of Science and Technology, Norway
[4] Machine Intelligence Research Labs - MIR Labs
ajith.abraham@ieee.org

Abstract- This paper describes an evolutionary clustering algorithm, which can partition a given dataset automatically into the optimal number of groups through one shot of optimization. The proposed method is based on an evolutionary computing technique known as the Bacterial Evolutionary Algorithm (BEA). The BEA draws inspiration from a biological phenomenon of microbial evolution. Unlike the conventional mutation, crossover and selection operaions in a GA (Genetic Algorithm), BEA incorporates two special operations for evolving its population, namely the bacterial mutation and the gene transfer operation. In the present context, these operations have been modified so as to handle the variable lengths of the chromosomes that encode different cluster groupings. Experiments were done with several synthetic as well as real life data sets including a remote sensing satellite image data. The results estabish the superiority of the proposed approach in terms of final accuracy.

Keywords: *Clustering, Pattern Recognition, genetic Algorithm, Bacterial Evolution, Metaheuristics.*

I. Introduction

The objective of clustering is to partition unlabeled data [1] into groups of similar objects. Each group, called a 'cluster', consists of objects that are similar between themselves and dissimilar to objects belonging to other groups. Clustering, or cluster analysis, is prevalent in any discipline that involves analysis of multivariate data. In the past few decades, cluster analysis has played a central role in diverse domains of science and engineering [1-3].

Clustering algorithms can be *hierarchical* or *partitional* [2]. In hierarchical clustering, the output is a tree showing a sequence of clustering with each cluster being a partition of the data set [2]. Partitional clustering algorithms, on the other hand, attempts to decompose the data set directly into a set of disjoint clusters. They try to optimize certain criteria (e.g. a Squared-error function).

The problem of partitional clustering has been approached from the diverse fields of knowledge like statistics (multivariate analysis) [4], graph theory [5], expectation maximization algorithms [6], artificial neural networks [7], evolutionary computing [8], swarm intelligence [9] and so on. While there remains a plethora of clustering algorithms, the important issue of determining the correct number of clusters in a virgin dataset is rarely touched upon. Finding the exact number of clusters is difficult because, unlike in supervised learning, there are no class labels for the data and, thus, no obvious criteria to guide the search. An account of the works undertaken in this direction can be found in [8].

In this work, the problem of automatic clustering has been approached from a framework of the Bacterial Evolutionary Algorithm (BEA) [10]. The BEA is a relatively new addition to the vast family of evolutionary computing techniques. Nature has served as an endless resource of ideas and metaphors for building the variations of the basic GA scheme [11]. Instead of using the conventional crossover, mutation and selection cycles of a GA, BEA employs two special operations to evolve its population of chromosomes (each of which encodes one trial solution of the optimization problem). The first of these is *bacterial mutation*, which mimics a process occurring in the bacterial genetics level and aims at improving the parts within the chromosomes. The second one, called *gene transfer operation*, is employed for the exchange of information between chromosomes in the population. It is motivated by the phenomenon of transfer of strands of genes through a population of bacteria. By means of this mechanism, one bacterium can rapidly spread its genetic information to the other cells without any crossover operation. Previously there have been only a few applications of the BEA mainly to model identification and fuzzy system parameters discovery [9]. This, to the best of our knowledge is the first paper that employs BEA to a pattern recognition problem.

The proposed algorithm, called by us the ACBEA (Automatic Clustering with BEA), encodes one complete partitioning of the data in a chromosome such that each part or gene in the chromosome represents a cluster. Since the number of clusters is not known a priori, the chromosomes come with variable lengths, each encoding a different number of classes in the data. We have incorporated a new operation called *chromosome repair* and also have modified both the mutation and gene transfer operations slightly to handle these variable-size chromosomes.

II. The Crisp Clustering Problem

A *pattern* is a physical or abstract structure of objects. It is distinguished from others by a collective set of attributes called *features*, which together represent a pattern [8]. Let P = {P_1, P_2... P_n} be a set of *n* patterns or data points, each having d features. These patterns can also be represented by a profile data matrix $\mathbf{X}_{n\times d}$ having *n* *d*-dimensional row vectors. The *i*-th row vector $\vec{X}_i$ characterizes the *i*-th object from the set P and each element $X_{i,j}$ in $\vec{X}_i$ corresponds to the j-th real value feature (*j* = 1, 2, ...,*d*) of the *i*-th pattern (*i* =1,2,...., *n*). Given such an $\mathbf{X}_{n\times d,}$ a partitional clustering algorithm tries to find a partition C = {C_1, C_2,......, C_K}of *K* classes, such that the similarity of the patterns in the same cluster is maximum and patterns from different clusters differ as far as possible. The partitions should maintain the following properties:

1. $C_i \neq \Phi \ \forall i \in \{1,2,...,K\}$.
2. $C_i \cap C_j = \Phi$, $\forall i \neq j$ and $i, j \in \{1,2,...,K\}$.
3. $$\bigcup_{j=1}^{K} C_j = P \qquad (1)$$

Since the given dataset can be partitioned in a number of ways maintaining all of the above properties, a fitness function (some measure of the adequacy of the partitioning) must be defined. The problem then turns out to be one of finding a partition $\mathbf{C}^*$ of optimal or near-optimal adequacy as compared to all other feasible solutions $\mathbf{C}$ = { C^1, C^2,........, $C^{N(n,K)}$} where,

$$N(n,K) = \frac{1}{K!}\sum_{i=1}^{K}(-1)^i \begin{pmatrix} K \\ i \end{pmatrix}^i (K-i)^i \qquad (2)$$

is the number of feasible partitions. This is same as,

$$\underset{C}{Optimize}\ f(X_{n\times d}, C) \qquad (3)$$

where *C* is a single partition from the set **C** and *f* is a statistical-mathematical function that quantifies the goodness of a partition on the basis of the distance measure of the patterns. It has been shown in [28] that the clustering problem is NP-hard when the number of clusters exceeds 3.

III. Automatic Clustering with BEA

In this Section, we describe the automatic clustering algorithm based on the BEA. The ACBEA starts with a population of variable length chromosomes, each of which encodes an entire grouping of the data. To measure the compactness, we first define the mean spread of the *i*-th cluster by taking the mean pair wise distance between the objects belonging to a cluster. This can be done by adding the elements of the upper or lower triangular portion of the adjacency matrix constructed for each cluster and dividing the sum by the total number of elements added i.e. $n_i(n_i - 1)/2$ where n_i is the number of objects belonging to the cluster under test. The mean spread of the i-th cluster can be mathematically formalized as:

$$MS_i = \frac{2}{n_i(n_i-1)}\sum_{k=1}^{n_i}\sum_{j=1}^{n_i} d(\vec{X}_k, \vec{X}_j) \qquad (4)$$

where $\vec{X}_k$ and $\vec{X}_j$ are two data points belonging to the same cluster C_i and $d(\vec{X}_i, \vec{X}_q)$ denotes their distance (similarity measure) in the feature space. At this point please note that we do not use here the popular criterion of intra-cluster variance, which squares the distance value and is more strongly biased towards the spherically shaped clusters around the cluster centroid. The compactness is defined as the reciprocal of the mean spread:

$$Com_i = \frac{1}{MS_i} \qquad (5)$$

3.1 Encoding the Chromosomes and Population Initialization

In order to make room for several possible choices of the number of clusters, ACBEA encodes an entire partitioning of the data in a single chromosome or bacterium cell. In each such chromosome, a gene encodes one cluster [12]. Thus if the *r*-th chromosome contains *K* genes {G_1^r,......,G_i^r,......,G_K^r}, it actually encodes a portioning of the dataset into *K* clusters. Each such group or cluster G_i^r of the *r*-th chromosome contains n_i^r number of data points such that $\sum_{i=1}^{K} n_i^r = n$ where *n* is the total number of data points in the given dataset. Each gene encoding a cluster actually consists of the integral labels of those data points, which belong to that cluster. For example, the *i*-th gene G_i^r of the *r*-th chromosome contains n_i^r integers $\{l_{i1}^r, l_{i2}^r,...., l_{ij}^r,....., l_{in_i^r}^r\}$ such that $l_{ij}^r \in \{1,....,n\}$ and l_{ij}^r physically means that the *j*-th object from the data set

P = {P$_1$, P$_2$... P$_n$} under test, belongs to the *i*-th cluster encoded by the *r*-th chromosome. This representation scheme has been shown diagrammatically in Figure 1.

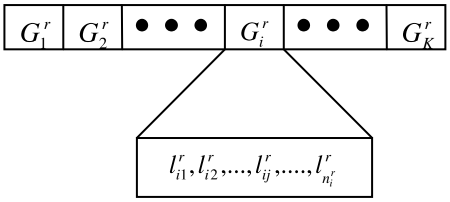

G_1^r
G_2^r
● ● ●
G_i^r
● ● ●
G_K^r
$l_{i1}^r, l_{i2}^r, \ldots, l_{ij}^r, \ldots., l_{n_i^r}^r$

FIGURE 1. CHROMOSOME REPRESENTATION SCHEME FOR ACBEA

In this work, we have refrained from using an encoding based on cluster centers even though these have been the most commonly found encoding scheme in the EA clustering literature. Initially each chromosome is generated in such a fashion that the number of clusters to be encoded by it is chosen randomly from the set of integers {2, 3..., K_{MAX}}. Hence the possible number of clusters may vary between 2 and K_{MAX} which may be supplied by the user.

3.2 Genetic Operators

1) Bacterial Mutation After the generation of the initial population, the bacterial mutation is applied to each chromosome one by one. The mutation operators of the ACBEA differ to some extent from those of the classical BEA reported in [10]. They are specifically designed to allow the length of the chromosomes to be changed dynamically as the evolutionary learning process progresses. In bacterial mutation, the first chromosome is chosen and then it is reproduced in m – 1 clone. Each clone is then mutated probabilistically with the one of the following three operators:

a) The *random replacement* mutation which occurs with a probability P_{rr}.

b) The *Split-gene* mutation occurring with a probability P_{sg}.

c) The *Merge-gene* mutation occurring with a probability of P_{mg}.

The clones are next evaluated according to some fitness function (to be discussed in the next section). The best chromosome from the m individuals is selected to remain in the population and the other *m*–1 individuals are deleted. This genetic operation is applied to all the chromosomes in the current population. Below we elaborate each mutation operator.

a)The random replacement mutation: In this operation, the *i*-th gene of a chromosome is selected randomly. A new one, not previously contained in the cluster encoded by that gene, with a probability P_{dr}, replaces each data label of the selected gene. This operation for the *j*-th data label of the *i*-th gene in *r*-th chromosome may be expressed as:

$$\textit{Replace with } l_k, \textit{ if rand (0, 1)} < P_{dr}$$

where l_k is an integer and $l_k \in \{1,\ldots,n\}$ but $l_k \notin \{ l_{i1}^r, l_{i2}^r, \ldots, l_{ij}^r, \ldots., l_{n_i^r}^r \}$, $\{l_{i1}^r, l_{i2}^r, \ldots, l_{ij}^r, \ldots., l_{n_i^r}^r\}$ is the set of data labels contained in the *i*-th gene and *rand* (0, 1) is a uniformly distributed random number in [0, 1].Thus the cluster encoded by the mutated gene is reconfigured, although the number of data points belonging to it remains unaltered.

b) The Split-gene mutation: In this case, we first rank the genes of a target chromosome in the ascending order of their cohesiveness. Next we set a gene selection probability P_{gs}. Based on P_{gs}, S_j number of genes with relatively low cohesion are selected for splitting, where S_j < K/2 and the minimum value of S_j is 1. Those selected may be represented as: $\{G_{s1}^r, \ldots G_{si}^r \ldots, G_{sj}^r\}$ and $G_{si}^r \in \{G_1^r, \ldots, G_i^r, \ldots, G_K^r\}$. Finally we randomly split each gene in $\{G_{s1}^r, \ldots G_{si}^r \ldots, G_{sj}^r\}$ into two clusters. The resulting number of genes in the chromosome becomes $K+2.S_j$ and this number has to be smaller than K_{MAX}, otherwise the mutation operator terminates.

c) The Merge-gene mutation: Like the previous case, we first set a gene selection probability P_{gm}. Based on P_{gm}, m_j number of genes with relatively low cohesion are selected for merging, where m_j < K/2 and the minimum value of m_j is 2. The selected genes in this way form a *merging pool*: $\{G_{m1}^r, \ldots G_{mi}^r \ldots, G_{mj}^r\}$ where m_j<K and $G_{mi}^r \in \{G_1^r, \ldots, G_i^r, \ldots, G_K^r\}$. Finally the genes in $\{G_{m1}^r, \ldots G_{mi}^r \ldots, G_{mj}^r\}$ are merged together into a single gene and is returned to a random location of the chromosome. In this way the length of the chromosome becomes $K - m_j + 1$ and this length must be greater than 2 (the minimum number of possible clusters) or otherwise the mutation operator terminates.

After mutating the *i*-th gene, ACBEA calls for a procedure for repairing the chromosomes. During the repairing procedure, all the genes of a mutated chromosome are scanned to remove duplicates in such a way that if a data label is found in another gene, it is removed. For those record labels that have not been assigned to any of the genes after their removal, they are randomly assigned to one of the genes (except the one from which they were removed).

2) The Gene Transfer Operation: The gene transfer operation is schematically illustrated in Figure 2. It takes place through the following steps:

a) The population is sorted in two halves: the half with individuals with better fitness is called the *superior half* while the other half is called the *inferior half*.

b) One chromosome is randomly picked up from the superior half and named *source chromosome*, while another one is selected at random from the inferior half and is named *destination chromosome*. Now according to some given criteria a ‘good’ part or gene from the source chromosome is to be transferred to the destination chromosome. Here, in context to the clustering problem, goodness of a cluster encoded by each gene is synonymous to its cohesiveness given by equations (4) and (5).

In ACBEA, the gene with highest compactness value is transferred from the source chromosome to the destination chromosome. In the destination chromosome, the coming gene overwrites a randomly selected gene with lower compactness value.

d) Steps 1 through 3 are repeated N_{inf} times where N_{inf} stands for the number of *infections* per generation.

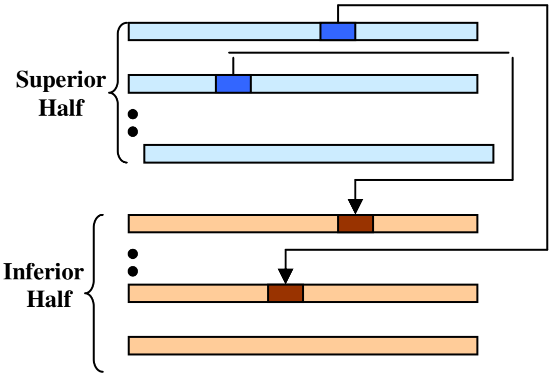


FIGURE 2. SCHEME FOR THE GENE TRANSFER OPERATION.

After the transfer of the gene to an inferior chromosome, it goes through a repairing mechanism, where all the genes of the chromosomes are thoroughly scanned to remove any duplicate data-label that occurs in more than one gene. The mutation and gene transfer operations are looped until a maximum number of generations have been exceeded. Flow-chart of the entire process has been provided in Figure 3.

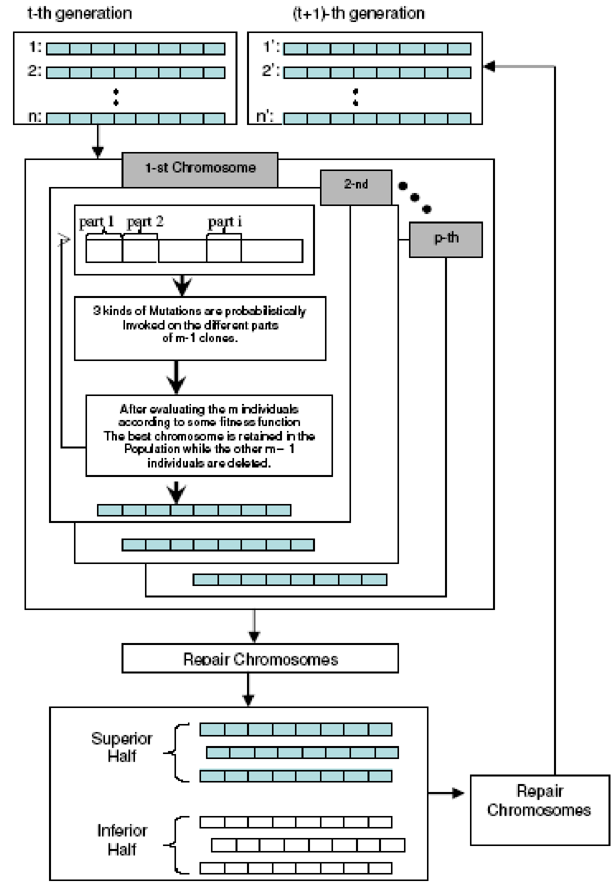

t-th generation
(t+1)-th generation
1:
2:
n:
1':
2':
n':
1-st Chromosome
2-nd
p-th
part 1
part 2
part i
3 kinds of Mutations are probabilistically Invoked on the different parts of m-1 clones.
After evaluating the m individuals according to some fitness function The best chromosome is retained in the Population while the other m – 1 individuals are deleted.
Repair Chromosomes
Superior Half
Inferior Half
Repair Chromosomes

FIGURE 3. FLOW-CHART FOR BEA-BASED CLUSTERING ALGORITHM.

3.3 Designing the Fitness Function

Selection of a suitable fitness function acts as a major driving force behind any evolutionary computing technique. The fitness function of the ACBEA is based on a recently developed *clustering validity index*. Cluster validity indices correspond to the statistical-mathematical functions used to evaluate the results of a clustering algorithm on a quantitative basis.

The fitness function of the ACBEA is based on the CS measure, recently proposed by Chou *et al.* [13]. Before applying the CS measure; centroid of a cluster is computed by averaging the data vectors belonging to that cluster using the formula

$$\vec{m}_i = \frac{1}{N_i} \sum_{\vec{X}_j \in C_i} \vec{X}_j \qquad (6)$$

A distance metric between any two data points $\vec{X}_i$ and $\vec{X}_j$ is denoted by $d(\vec{X}_i, \vec{X}_j)$. Then the CS measure can be defined as,

$$CS(K) = \frac{\frac{1}{K}\sum_{i=1}^{K}[\frac{1}{N_i}\sum_{\vec{X}_i \in C_i} \max_{\vec{X}_q \in C_i}\{d(\vec{X}_i, \vec{X}_q)\}]}{\frac{1}{K}\sum_{i=1}^{K}[\min_{j \in K, j \neq i}\{d(\vec{m}_i, \vec{m}_j)\}]}$$

$$= \frac{\sum_{i=1}^{K}[\frac{1}{N_i}\sum_{\vec{X}_i \in C_i} \max_{\vec{X}_q \in C_i}\{d(\vec{X}_i, \vec{X}_q)\}]}{\sum_{i=1}^{K}[\min_{j \in K, j \neq i}\{d(\vec{m}_i, \vec{m}_j)\}]} \qquad (7)$$

A lower value of CS (*K*) indicates the better quality of the partitioning. As can easily be perceived, this measure is a function of the ratio of the sum of within-cluster scatter to between-cluster separation. The CS measure based fitness function (to be maximized) for the *i*-th chromosome encoding *K* classes may be described as:

$$f_1 = \frac{1}{CS_i(K) + eps} \qquad (8)$$

where *eps* is a small bias term equal to 0.002 and it prevents the fitness function from moving up to infinity if somehow CS_i *(K)* becomes zero for a chromosome. We choose CS measure as the basis of the fitness evaluation mechanism, as according to the authors, it is more efficient in tackling clusters of different densities and/or sizes than the other popular validity measures, the price being paid in terms of high computational load with increasing *K* and *n*.

IV. Experimental Setup

This section describes the setup for the simulation experiments undertaken to compare the performance of ACBEA with three other competitive clustering algorithms using five benchmark datasets of varying levels of complexity.

4.1 The Datasets Used

We have used five real life datasets to compare the contestant clustering methods. The following real-life datasets [16] are used in this study. Here *n* = number of data points, *d* = number of features and *K* = number of clusters.

1) Iris plants database (*n* =150, *d* = 4, *K* = 3)

2) Glass (*n* =214, *d* = 9, *K* = 6)

3) Breast cancer data set (*n* = 683, *d* = 9, *K* = 2)

4) Wine (*n* = 178, *d* = 13, *K* = 3)

5)Vowel Dataset (*n* = 871, *d* = 3, *K* = 6)

4.2 Algorithms for Comparisons

In order to evaluate the performance of BEA, we compare it with two recently developed automatic clustering algorithms known as the DCPSO (Dynamic Clustering with Particle Swarm Optimization) [15] and GCUK (Genetic Clustering with an Unknown *K*). We also take into account the classical k-means algorithm [17] for the comparative study.

4.3 Algorithmic Settings

This paper uses the "usual" parameter settings for the three search techniques. Table 1 summarizes these settings. In addition, the same parameter settings have been used for all the clustering problems to make the comparison fair. In Table 1, Popsize indicates the size of the population.

Table 1: Parameters for competitive clustering algorithms

GCUK		DCPSO		ACBEA	
Para-meter	Value	Para- meter	Value	Para- meter	Value
Pop_size	50	Pop size	100	Pop_size	20
Cross-over Prob-ability μ_c	0.8	Inertia Weight	0.72	P_{rr}	0.54
				P_{sg}	0.31
Muta-tion prob-ability μ_m	0.001	C_1, C_2	1.494	P_{mg}	0.23
		P_{ini}	0.75	P_{dr}	0.30
				P_{gm}	0.44
K_{max} K_{min}	20 2	K_{max} K_{min}	20 2	K_{max} K_{min}	20 2

4.4 Simulation Strategy

All the three optimization algorithms are stochastic in nature. Hence, for each problem they have been run several times. Since the algorithms have been made to work on the same framework as described in section 4, the actual number of classes has not been supplied to any of them for any of the problems detailed in section 5.Out of multiple runs for a given algorithm for a given problem, only those were considered successful in which the algorithm could determine the true number of classes. The results have been stated in terms of the mean values and standard deviations over 50 successful runs in each case. Since the population size for DCPSO, PSO and GA are markedly different; the present authors choose number of fitness evaluations as a measure of computation time instead of 'generations' or 'iterations'. Number of fitness evaluations roughly equals the product of the population size and the number of generations. In this study, all the competitor algorithms except the K-means were allowed to run for 2 × 10^5 function evaluations. The algorithms were compared based on:

1) The *CS measure* as defined in equation (7);

2) The *intra-cluster distances*, i.e. the distance between data vectors within a cluster, where the objective is to minimize the intra-cluster distances.

3) The *inter-cluster distances*, i.e. the distance between the centroids of the clusters, where the objective is to maximize the distance between clusters.

4) The *robustness,* i.e. the number of times an automatic clustering algorithm can find out the correct number of clusters.

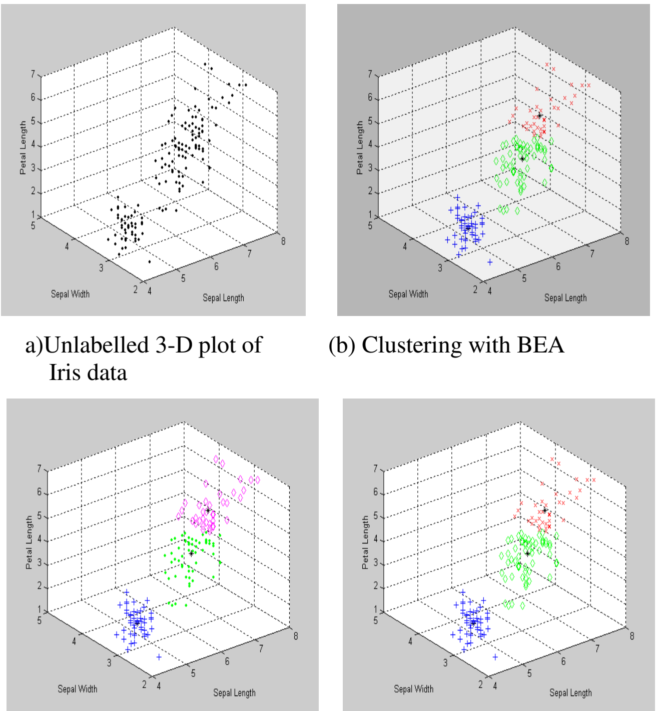
The latter two objectives respectively correspond to crisp, compact clusters that are well separated. All the algorithms discussed here have been developed from scratch in Visual C++ platform on a Pentium IV, 2.2 GHz PC, with 512 KB cache and 2 GB of main memory in Windows Server 2003 environment. The graphs and figures have been obtained using MATLAB 6.5.

V. Empirical Results

Table 2 summarizes the clustering result over five public domain datasets in terms of the final CS-measure value, mean inter-cluster, and intra-cluster distances. For the population based search algorithms, the mean best value and the standard deviation over 50 successful runs have been reported. Except for the K-means algorithm, no method has been supplied with the exact number of classes. Only a maximum number of clusters (20 for all cases) has been specified for all of them. Figure 4 provides a visual feel of the performance of fourclustering methods over the iris dataset. The actual four-dimensional dataset has been plotted in 3 dimensions using the first three features only.

Row 1 of Table 2 and Figure 4 indicate the fact that Fisher's iris dataset is not sufficiently challenging to compare the performance between advanced clustering algorithms despite its great popularity in the clustering community. All the three methods performed equally well on this dataset whereas K-means algorithm frequently got stuck in local minima. Substantial performance differences occur for challenging clustering problems with a large number of data-items and clusters as well as overlapping cluster shapes. Over the wine dataset, accuracy of K-means and the new method where comparable, however this was the only case where PSO based clustering yielded best results. Over the rest three datasets and especially for the vowel data, that has strongly overlapping classes associated with it, the BEA method gives best results in terms of robustness and accuracy. It can be seen that in almost all the cases BEA based method gives a greater number of successful runs.

Since all the datasets used here have their nominal partitions known to the user, the present work also computes the mean number of misclassified data-points. This is the average number of objects that were assigned to clusters other than according to the nominal classification. Table 4 reports the corresponding mean values and standard deviations over 50 successful runs for all the three population based algorithms. One can easily see that the DCPSO based dynamic clustering algorithm yields lowest number of misclassified items in most of the cases. However, for breast cancer dataset, performance of the K-means is comparable to the new method.



a)Unlabelled 3-D plot of Iris data

(b) Clustering with BEA

(c) ClusteringwithGCU

(d)Clustering with DCPSO

FIGURE 4: PERFORMANCE OF FOUR CLUSTERING METHODS OVER THE IRIS DATASET

A non-parametric statistical significance test called Wilcoxon's rank sum test for independent samples [18, 19] has been conducted at the 5% significance level on the CS measure data of Table 2. Table 3 shows reports the *P*-values produced by Wilcoxon's rank sum test for comparison of the error scores of two groups (one group corresponding to BEA and the other corresponding to a competitor algorithm) at a time. As a null hypothesis, it is assumed that there is no significant difference between the mean values of two groups. Whereas, the alternative hypothesis is that there is significant difference in the mean values of the two groups. All the *P*-values reported in the Table are less than 0.05 (5% significance level). This is strong evidence against the null hypothesis, indicating that the better mean values of the performance metrics produced by BEA is statistically significant and has not occurred by chance.

Table 2. Clustering result over five real life datasets (For alalgorithms, mean and standard deviation over 50 successful runs and equal number of Function Evaluations 2×10^5 has been used for the last three Algorithms)

Dataset	Algorithm	successful runs %	CS measure	Intra cluster Distance	Inter cluster Distance
Iris	K-means	NA	0.9562±0.06021	3.542±0.056	1.726±0.052
	BEA	**88%**	**0.1735±0.029**	**3.125±0.068**	**2.487±0.063**
	DCPSO	40%	0.6508±0.073	3.144±0.056	2.0144±0.0474
	GCUK	48%	0.9081±0.1267	3.274±1.72	2.0058±0.561
Wine	K-means	NA	1.2645±0.0021	3.975±0.0026	0.7113±0.00054
	BEA	**86%**	**0.8065±0.0037**	**3.112±0.02573**	**0.8915±0.00663**
	DCPSO	60%	0.7721±0.046	3.851±0.0173	0.9613±0.00054
	GCUK	54%	1.3945±0.0281	3.065±0.1226	0.7563±0.0078
Breast-Cancer	K-means	NA	0.6445±0.041	6.975±0.0026	1.7113±0.00054
	BEA	**68%**	**0.4265±0.0035**	**5.0212±0.02573**	**1.2215±0.03663**
	DCPSO	22%	0.6021±0.32	5.851±0.0173	2.9613±0.00054
	GCUK	30%	0.548±0.0478	6.851±1.0161	2.4413±0.0194
Vowel	K-means	NA	2.32±0.0047	149962.75±0.0026	1898.7113±0.0054
	BEA	**74%**	**0.9265±0.075**	**142809.12±0.02573**	**2982.8215±0.0663**
	DCPSO	56%	1.1321±0.231	15002.51±0.0173	1932.9613±0.00054
	GCUK	26%	2.8921±0.62	149987.34±0.0923	2009.903±0.0173
Glass	K-means	NA	0.92±0.68	11054.561±309.4	1053.890±22.14
	BEA	**80%**	**0.7065±0.0062**	**10974.644±210.1**	**954.672±19.04**
	DCPSO	66%	1.1089±0.29	11088.546±333.9	1031.872±34.95
	GCUK	48%	1.0977±0.112	11109.302±435.6	1077.929±18.77

Table 3. *P*-values produced by Wilcoxon's rank sum test taking comparing BEA with other algorithms on the CS-measure data of Table 2.

Dataset	*P*-Value		
	DCPSO	GCUK	k-means
Iris	1.8267e-004	1.8165e-004	4.9367e-005
Wine	1.4623e-004	1.8143e-005	3.8264e-005
Breast Cancer	1.9252e-004	1.7446e-004	1.3573e-004
Vowel	1.6924e-004	1.8001e-004	1.8115e-004
Glass	1.8824e-004	1.8179e-004	1.8235e-004

Table 4. Mean Classification Error over nominal partition and standard deviation over the 50 successful runs on the real world datasets for the four algorithms compared

Dataset	Mean Classification Error			
	BEA	DCPSO	GCUK	k-means
Iris	**2.35±0.00**	4.15±0.0	5.00±0.00	3.96±0.00
Wine	**36.65±0.0**	99.4±1.09	100.24±1.05	114.50±1.53
Breast Cancer	**22.25±0.28**	27.01±1.25	29.00±1.55	29.15±0.50
Vowel	**418.75±3.10**	453.58±6.61	476.42±6.92	473.72±4.25
Glass	**92.55±0.19**	102.1±0.68	98.21±0.08	105.36±0.54

VI. CONCLUSIONS

This paper has presented a new, Bacterial Evolutionary Algorithm-based strategy for crisp clustering of real world datasets. An important feature of the proposed technique is that it is able to find the optimal number of clusters automatically (that is, the number of clusters does not have to be known in advance) even for very high dimensional datasets where tracking of the number of clusters may be well nigh impossible. The proposed ACBEA algorithm has been shown to meet or beat two other state-of-the-art clustering algorithms in a statistically meaningful way over majority of the benchmark datasets discussed here. This certainly does not lead us to claim that ACBEA may outperform DCPSO or GCUK over every dataset since it is impossible to model all the possible complexities of a real life data with the limited test-suit that we used for testing the algorithms. In addition, the performance of DCPSO and GCUK may also be enhanced with a judicious parameter tuning, which renders itself to further research with these algorithms. However, the only conclusion we can draw at this point is that BEA with the suggested modifications can serve as an attractive alternative for dynamic clustering of completely unknown datasets.

REFERENCES

1. R. O. Duda , P. E. Hart., Pattern Classification and Scene Analysis. John Wiley and Sons, 1973.

2. A. K. Jain, M. N. Murty, P. J. Flynn, Data clustering, a review, ACM Computing Surveys, vol. 31, no.3, 264—323, (1999).

3. R. Xu and D. Wunsch, Clustering, Wiley-IEEE book series on Computational Intelligence, 2008.

4. E.W. Forgy, Cluster Analysis of Multivariate Data: Efficiency versus Interpretability of classification, Biometrics, 21, (1965) 768-9.

5. C. T. Zahn, Graph-theoretical methods for detecting and describing gestalt clusters. IEEE Transactions on Computers C-20 (1971), 68–86.

6. M.H. Law, M.A.T. Figueiredo, and A.K. Jain, Simultaneous Feature Selection and Clustering Using Mixture Models, IEEE Transactions Pattern Analysis and Machine Intelligence, vol. 26, no. 9, pp. 1154-1166, Sept. 2004.

7. N. R. Pal, J. C. Bezdek, E. C.-K. Tsao, Generalized clustering networks and Kohonen's self-organizing scheme, IEEE Trans. Neural Networks, vol 4, (1993) 549–557.

8. S. Das, A. Abraham, and A. Konar, Metaheuristic Clustering, Studies in Computational Intelligence, Springer Verlag, Germany, 2009.

9. A. Abraham, S. Das, and S. Roy, Swarm Intelligence Algorithms for Data Clustering, Soft Computing for Knowledge Discovery and Data Mining, O. Maimon and L. Rokach (Eds.), Springer Verlag, Germany, pp. 279-313, 2007.

10. N. E. Nawa, T. Furuhashi, Fuzzy System Parameters Discovery by Bacterial Evolutionary Algorithm, in IEEE Transactions on Fuzzy Systems, vol. 7, no. 5, pp.608-616, October 1999.

11. J.H. Holland, Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Harbor (1975).

12. E. Falkenauer, Genetic Algorithms and Grouping Problems, John Wiley and Son, Chichester (1998).

13. C.H. Chou, M.C. Su, E. Lai, A new cluster validity measure and its application to image compression. Pattern Analysis and Applications 7(2), (2004) 205-220.

14. S. Bandyopadhyay, U. Maulik, Genetic clustering for automatic evolution of clusters and application to image classification, Pattern Recognition, 35, (2002) 1197-1208.

15. M. Omran, A. Salman and A. Engelbrecht. Dynamic Clustering using Particle Swarm Optimization with Application in Unsupervised Image Classification. Fifth World Enformatika Conference (ICCI 2005), Prague, Czech Republic, 2005.

16. C. Blake, E.Keough and C.J.Merz, UCI repository of machine learning database (1998). http://www.ics.uci.edu/~mlearn/MLrepository.html

17. J. MacQueen, Some methods for classification and analysis of multivariate observations, Proceedings of the Fifth Berkely Symposium on Mathematical Statistics and Probability, 281-297 (1967).

18. F. Wilcoxon, Individual comparisons by ranking methods. Biometrics, 1, 80-83, 1945.

19. S. García, D. Molina, M. Lozano, and F. Herrera, A Study on the Use of Non-Parametric Tests for Analyzing the Evolutionary Algorithms' Behaviour: A Case Study on the CEC'2005 Special Session on Real Parameter Optimization, Journal of Heuristics, DOI: 10.1007/s10732-008-9080-4, in press (2008).