The New Window Density Function for Efficient Evolutionary Unsupervised Clustering

Dimitris K. Tasoulis
Computational Intelligence Laboratory,
Department of Mathematics, University of Patras,
GR-26110 Patras, Greece.
dtas@math.upatras.gr

Michael N. Vrahatis
Computational Intelligence Laboratory,
Department of Mathematics, University of Patras,
GR-26110 Patras, Greece.
vrahatis@math.upatras.gr

Abstract- Evolutionary clustering is a recent trend in cluster analysis, that has the potential to yield high partitioning accuracy results. Traditional evolutionary techniques applied in clustering are typically hindered by the high cost involved in the computation of the objective function. In this paper we propose a novel objective function, that can provide fitness function values in sublinear time. Next we develop an evolutionary scheme, to evolve cluster solutions and demonstrate how the number of clusters can be estimated from the final result. Finally, by employing real world datasets, we exhibit the high quality clustering results that this sceme can provide.

1 Introduction

Clustering is a fundamental step in the process of transforming data to knowledge. It aims at discovering groups (clusters) in a set of objects such that similarity among the objects in the same group is higher than that of objects belonging to different clusters.

The first references to clustering date as back as the fourth century B.C. by Aristotle and Theophrastos, but it was not until 1939, that one of the first comprehensive foundations of these methods was published [30].

The application domain of clustering techniques is very wide including data mining [14], text mining [10, 13], statistical data analysis [2], compression and vector quantization [23], global optimization [6, 29] and web personalization [24].

Clustering is a hard problem, since even the simplest clustering problems are known to be NP-Hard [1]. The Euclidean k-center problem [1] in the plane is NP-Hard [20]. In fact, it is NP-Hard to approximate the two-dimensional k-center problem even under the L_∞-metric [20].

Clustering algorithms are traditionally categorized into three main categories, Hierarchical, Partitioning [26] and Distance-based. Hierarchical clustering algorithms construct hierarchies of clusters in a top-down (agglomerative) or bottom-up (divisive) fashion. Hierarchical clustering algorithms have proved to yield high quality results especially for applications involving clustering text collections. None the less, their high computational requirements, usually limits their applicability in real life applications, where the number of samples and their dimensionality is typically high (the cost is quadratic to the number of samples).

Partitioning clustering algorithms, start from an initial clustering (that can be randomly formed) and create partitionings by iteratively adjusting the clusters based on the distance of the data points from a representative member of the cluster. The most commonly used partitioning clustering algorithm is k-means. This algorithm initializes k centers and iteratively assigns each data point to the cluster whose centroid minimizes the euclidean distance from the data point. Algorithms of this type can give good clustering results at low cost, since their running time is proportional to kn, where n is the number of patterns present in the dataset. However, their results rely heavily on their initialization and they can converge to arbitrary local optima.

Distance based clustering algorithms create a partitioning by considering neighbors of data points. DBSCAN [25] is a distance-based clustering algorithm that has proved quite effective for spatial databases. Clusters are considered as high density neighborhoods of data points. Although the density parameter is critical for the successful application of DBSCAN, recently proposed heuristics appear to yield high quality results. The computational complexity of DBSCAN comes up to $O(n \log(n))$ under the assumption that the data are organized in a spatial index (R^*-tree).

In evolutionary clustering, a solution to the clustering problem is typically encoded as a chromosome. Next, by employing evolutionary operators and a population of solutions the algorithm probes the search space to find a globally optimum partition of the data. The most commonly used evolutionary operators are: selection, recombination, and mutation. Each operator transforms one or more input chromosomes into one or more output chromosomes. A fitness function evaluated on a chromosome determines a chromosome's likelihood of surviving into the next generation. In early approaches [8, 18], chromosomes encoded the partition of n objects into K clusters and *Genetic Algorithms* were employed to identify the best partition. However, the sensitivity of GAs to the selection of the various parameters like population size, and crossover and mutation probabilities, as well as, the difficulties associated with the representation scheme, presented a major problem. Better results were obtained through hybrid approaches [4].

However, it is possible to represent the clustering procedure as an optimization problem of locating the optimal centroids of clusters. Thus, all evolutionary techniques can be employed since a clustering solution can be represented as a real-valued vector of the centroids. Previous approaches employed *Evolutionary Strategies* [5], Evolutionary Programming [15], and recently *Particle Swarm Opti-*

mization [31]. All these approaches demonstrated that it is possible to obtain high quality partitions, but at a high computational cost.

In this paper we attempt to tackle the high computational cost of traditional evolutionary techniques by introducing a new fitness criterion. This criterion is based on a windowing technique already employed in other clustering algorithms [28, 32]. The proposed approach is independent of the evolutionary technique employed. In this paper we employ the *Differential Evolution* algorithm as recent work [21] has demonstrated its superior performance on such problems.

A critical and open issue in cluster analysis, is the determination of the number of clusters present in a dataset. The evolutionary clustering techniques proposed so far, with the exception of [17], require from the user to specify the number of clusters present in the data prior to the execution of the algorithm. The proposed approach can provide an approximation to the number of clusters present in a dataset.

The rest of the paper is organized as follows. In Section 2, we analyze the proposed fitness function. Next, in Section 3 we describe the proposed evolutionary scheme and in Section 4 we present experimental results that demonstrate the applicability of the proposed approach. The paper ends with concluding remarks and discussion in Section 5.

2 Window Density Function

Let the data set comprise a set $X = \{x_1, \ldots, x_n\}$, where x_j is a data vector in the d dimensional Euclidean space $\mathbb{R}^d$. A k clustering of X is a partition C of X into k disjoint groups C_i, for $i = 1, 2, \ldots, k$. The clustering problem amounts to the determination of a partition of X which is optimal with respect to a function f that quantifies the goodness of the partition.

Different statistical functions have been proposed for f [19, 33]. But in all previous approaches at least a full scan over the dataset is necessary to compute the function value for a specific instance. Evolving a population using such a fitness criterion can be expensive in terms of computational cost, compared to k-means like approaches that typically do not require more than 10 to 20 scans of the dataset.

In the present contribution we propose the *Window Density Function* (WDF) that overcomes the aforementioned limitations:

Definition 1: Let a d-range of size $a \in \mathbb{R}$ and center $z \in \mathbb{R}^d$ be the orthogonal range $[z_1 - a, z_1 + a] \times \cdots \times [z_d - a, z_d + a]$. Assume further, that the set $S_{a,z}$, with respect to the set X, is defined as:

$$S_{a,z} = \{y \in X : z_i - a \leqslant y_i \leqslant z_i + a, \quad \forall i = 1, \ldots, d\}.$$

Then the *Window Density Function* WDF for the set X, with respect to a given size $a \in \mathbb{R}$, is defined as:

$$\text{WDF}_a(z) \quad = \quad |S_{a,z}|. \tag{1}$$

In other words, WDF represents the number of points from the dataset X, that reside in a window of size a centered at z. WDF is a meaningful clustering objective function, since as the center of a d-range, z, moves to the center of the cluster the number of points around it should increase. As it is obvious the size a, is critical to the procedure as it determines the location of the minima of the objective function. To illustrate this we employ the dataset $Dset_1$, exhibited in Fig. 1. This dataset contains 500 points organized in 5 clusters with 100 points each. Each cluster is constructed by sampling 100 points from a two dimensional Gaussian distribution. The mean of each distribution was randomly scattered in the range $[0, 200]^2$, and the covariance matrices were randomly generated by obtaining for each element of the matrix a random number between 1 and 2.

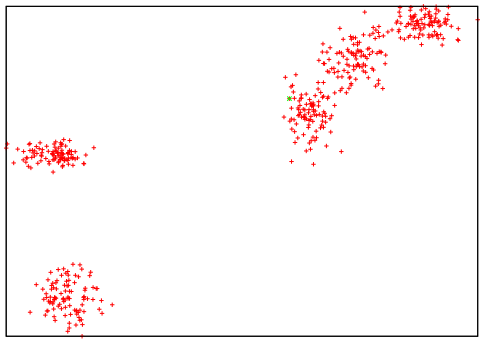

Figure 1: Dataset $DSet_1$

In Fig. 2, the $3d$-plots of WDF are provided to visualize the impact of the parameter a. As the value of a increases, the extreme points of WDF tend to merge. When $a = 1$ there are five maxima, equal to the number of clusters. On the other hand, when $a = 10$, the three maxima corresponding to the three closest clusters previously identified merge to a single one.

The most important feature of the proposed density function is that it is not necessary to scan the entire dataset to obtain a fitness value for a specific object. In particular, the computation of WDF is the well studied Computational Geometry *Orthogonal Range Search Problem*. Numerous Computational Geometry techniques have been proposed to address this problem. All these techniques employ a preprocessing stage at which they construct a data structure storing the patterns. This data structure allows them to answer range queries fast. In Table 1 the computational complexity of various such approaches is summarized. In detail, for applications of very high dimensionality, data structures like the Multidimensional Binary Tree [22], and Bentley and Maurer [7] seem more suitable. On the other hand, for low dimensional data with a large number of points the approach of Alevizos [3] appears more attractive.

3 Evolutionary Clustering under the WDF Objective Function

Evolutionary algorithms (EAs) have their roots in the stochastic search methods scientific domain, and try to mimic the natural biological evolution process. Utilizing the principle of survival of the fittest they try to evolve an

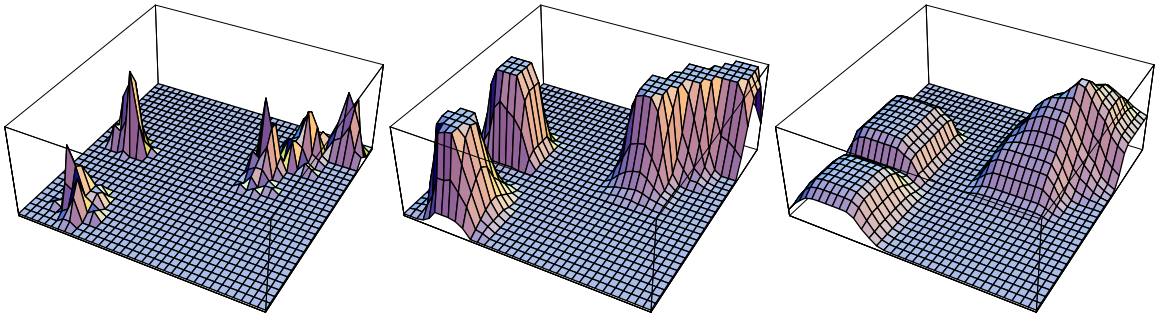

Figure 2: 3D-Plot of WDF for $Dset_1$ and $a = 1, 5, 10$ from left to right

	Preprocessing		
Method	**Time**	**Space**	**Query time**
Multidim. Binary Tree [22]	$\theta\left(dn\log n\right)$	$\theta\left(dn\right)$	$O\left(s+dn^{1-1/d}\right)$
Range Tree [22]	$O\left(n\log^{d-1}n\right)$	$O\left(n\log^{d-1}n\right)$	$O\left(s+\log^{d}n\right)$
Wilard and Lueeker [22]	$O\left(n\log^{d-1}n\right)$	$O\left(n\log^{d-1}n\right)$	$O\left(s+\log^{d-1}n\right)$
Chazelle [11]	$O\left(n\log^{d-1}n\right)$	$O\left(n\frac{\log^{d-1}n}{\log\log n}\right)$	$O\left(s+\log^{d-1}n\right)$
Chazelle and Guibas [12]	$O\left(n\log^{d+1}n\right)$	$O\left(n\log^{d}n\right)$	$O\left(s+\log^{d-2}n\right)$
Alevizos [3]	$O\left(n\log^{d-1}n\right)$	$O\left(n\log^{d-1}n\right)$	$O\left(s+\log^{d-2}n\right)$
Bentley and Maurer [7]	$O\left(n^{2d-1}\right)$	$O\left(n^{2d-1}\right)$	$O\left(s+d\log n\right)$
Bentley and Maurer [7]	$O\left(n^{1+\varepsilon}\right)$	$O\left(n^{1+\varepsilon}\right)$	$O\left(s+\log n\right)$
Bentley and Maurer [7]	$O\left(n\log n\right)$	$O\left(n\right)$	$O\left(n^{\varepsilon}\right)$

Table 1: Methods for orthogonal range search.

initial population of potential solutions to obtain a globally optimal result. In this paper, from the broad field of EAs we employ Differential Evolution [27]. Of course our approach can be applied using any other EA.

DE evolves the *population* of potential solutions (individuals), using three operators, *mutation*, *recombination* and *selection*. An individual, in the clustering context, is expressed using a predetermined number of d–dimensional vectors that represent the centers of the d–ranges, which in turn constitute the clustering result. The fitness of each individual is measured by the sum of the WDF function over all the d–ranges, under a fixed value of the parameter a. The remaining procedure of the DE algorithm remains unchanged.

At the first step, all individuals are initialized using a random number generator. At the mutation step, for each $i = 1, 2, \ldots, P$ (P represents the size of the population) a new mutant weight vector v^i_{g+1} is generated by combining vectors, randomly chosen from the population, and exploiting one of the mutation operators (2)–(6):

$$v^i_{g+1} = \omega^{best}_g + \mu(\omega^{r1}_g - \omega^{r2}_g), \quad (2)$$

$$v^i_{g+1} = \omega^{r1}_g + \mu(\omega^{r2}_g - \omega^{r3}_g), \quad (3)$$

$$v^i_{g+1} = \omega^{i}_g + \mu(\omega^{best}_g - \omega^{i}_g) + \mu(\omega^{r1}_g - \omega^{r2}_g), \quad (4)$$

$$v^i_{g+1} = \omega^{best}_g + \mu(\omega^{r1}_g - \omega^{r2}_g) + \mu(\omega^{r3}_g - \omega^{r4}_g), \quad (5)$$

$$v^i_{g+1} = \omega^{r1}_g + \mu(\omega^{r2}_g - \omega^{r3}_g) + \mu(\omega^{r4}_g - \omega^{r5}_g), \quad (6)$$

where ω^{r1}_g, ω^{r2}_g, ω^{r3}_g, ω^{r4}_g and ω^{r5}_g are randomly selected vectors, different from ω^i_g, ω^{best}_g is the best member of the current generation. Finally, the positive mutation constant μ controls the magnification of the difference between two weight vectors. For the rest of the paper, we refer to the differential evolution algorithm that uses Eq. (2) as the mutation operator with DE_1 , DE_2 for the algorithm that uses Eq. (3), and so on.

Having constructed the mutant vectors, the recombination step is initiated that constructs a trial vector for each individual. At the recombination step, for each component $j = 1, 2, \ldots, L$ of the mutant vector a random number $r \in [0, 1]$ is generated. If r is smaller than the predefined recombination constant p, the j-th component of the mutant vector v^i_{g+1} becomes the j-th component of the trial vector. Otherwise, the j-th component of the target vector is selected as the h–th component of the trial vector. Finally, the resulting trial vector replaces the initial individual if it yields a better WDF function value. Otherwise it is discarded. This constitutes the selection step.

As it is obvious the evolutionary optimization procedure described above aims at discovering the set of d-ranges that include as many points from the dataset as possible. Thus a single execution is able to determine a clustering result. Note that in the final clustering solution empty d-ranges may appear, or even d-ranges that overlap may exist. By employing the *merge* operation of the unsupervised

k-windows clustering algorithm [28], the number of clusters can be approximated. During this step, for each pair of overlapping windows, the number of patterns that lie in their intersection is computed. With respect to the proportion of this number to the total number of points contained in each window, the algorithm can decide whether to either:

(a) Ignore one window if the proportion is very high.

(b) Consider the windows to contain parts of the same cluster if the proportion is relatively high.

(c) Consider the windows to capture different clusters, if the proportion is low.

An example of this operation is exhibited in Fig. 3.

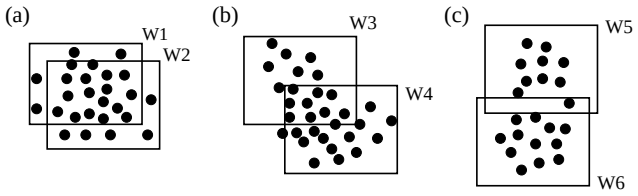

Figure 3: (a) W_1 and W_2 satisfy the similarity condition and W_1 is deleted. (b) W_3 and W_4 satisfy the merge operation and are considered to belong to the same cluster. (c) W_5 and W_6 have a small overlapment and capture two different clusters.

A high level description of the proposed algorithmic scheme follows:

DE Unsupervised Clustering (DEUC)
Construct a data structure for the storing of the data.
Set the parameter a of WDF function.
Repeat
  Execute the DE algorithm.
Until a sufficient part of the dataset is covered
  or a maximum number or iterations is performed.
Merge the resulting d–ranges
Report the final clusters.

4 Experimental Results

To demonstrate the applicability of the proposed approach we firstly employ $Dset_1$, exhibited in Fig. 1, which is two dimensional and allows the visual inspection of the results. Note that in all the experiments reported in this section the population size was set to 20 individuals, and a maximum of 200 epochs was allowed. The DE parameters μ and p were set to 0.6, and 0.8, respectively, in all experiments. Moreover, if the d–ranges of the best individual discovered contain more than 90% of the total points the execution of DE terminates. The application of the DEUC algorithm over the $Dset_1$ dataset with the parameter a obtaining values $1, 2, 5, 10$ is exhibited in Fig. 4. These results were obtained, by stopping the iterative executions of DE when more than 90% of the dataset was covered. Each individual encoded the center of five d-ranges. Comparing the clustering result, with the 3D-plots of the WDF function in

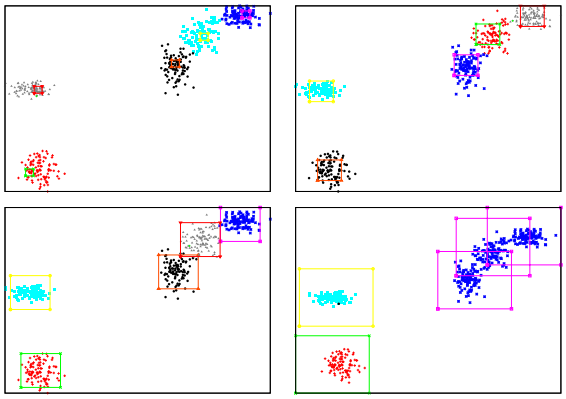

Figure 4: The clustering result of DEUC for $a = 1, 3, 5, 10$

Fig. 2, it is obvious that DEUC is able to detect the extrema of WDF and form a clustering result that is in accordance with the form of WDF. The colors in the plots correspond to the different cluster labels of the points that were assigned to the closest d-range under the Euclidean metric. It is obvious that DEUC is able to provide visually optimal clustering results when a ranges between 1 and 5. On the other hand, when a is too large the adjacent clusters are merged to a single cluster by the merging procedure.

Comparing the results of DEUC involves the usage of a clustering algorithm that can approximate the number of clusters. To compare the results of DEUC with other approaches we employ the DBSCAN clustering algorithm [25]. This choice is motivated by the fact that DBSCAN computes the number of points ($MinPts$) that reside in a hypersphere of size Eps. Thus, the Eps parameter of DBSCAN is strongly related to the a parameter of the WDF function. The execution of DBSCAN on $Dset_1$, setting $MinPts = 5$, (anything with less than 5 points in an Eps neighborhood around it is considered noise), and for Eps obtaining the values $Eps = 1, 3, 5, 10$ is exhibited in Fig. 5. Similarly in this case the colors designate different cluster labels, and the red crosses represent points recognized as noise. From the plots we can see that DBSCAN is more sensitive to the value of Eps than DEUC is on the value of a. Moreover, for DBSCAN to be able to recognize the three different adjacent clusters a very careful selection of Eps and $MinPts$ is needed.

Next, in Fig. 6, we investigate the ability of DEUC to approximate the number of clusters. To this end we apply DEUC using 3, 5, 10 and 15 windows. As illustrated, when the number of d-ranges is less than the true number of clusters, each d-range is located over a minimum of WDF, but due to the inability to cover all the minima the cluster labels are incorrect. On the other hand, as the number of d-ranges becomes larger than the real number of clusters, the algorithm has no problem of detecting the five clusters, since the merging procedure assigns correctly the cluster labels.

The complexity of the DEUC algorithm can be analyzed by the number of function evaluations it requires to provide a clustering result. As already mentioned for each function evaluation a range search operation over the dataset is per-

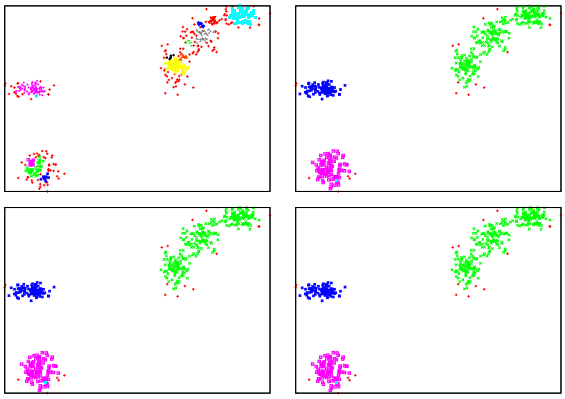

Figure 5: The clustering result of DBSCAN for $Eps = 1, 3, 5, 10$

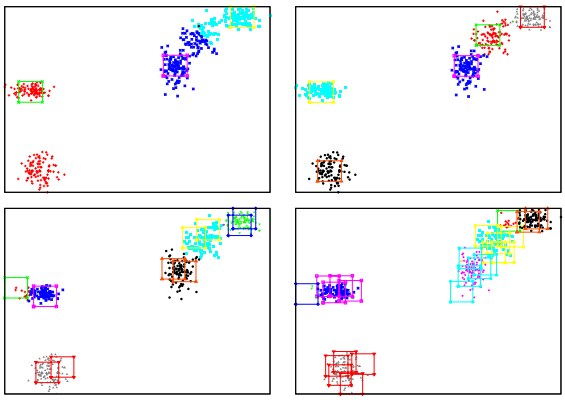

Figure 6: The impact on the clustering result of different number of d-ranges (3,5,10 and 15) when $a = 3$

formed. Measuring the total number of range searches that are needed is an indication of the relative speed of DEUC. To this end, we constructed $Dset_2$ in a manner similar to $Dset_1$, but with a size ranging from 5000 to 30000 points. The mean number of range searches required over 100 executions of DEUC, for all mutation operators is depicted in Fig. 7. From this figure it is clear that all the DE operators require a steady number of range searches to converge, irrespective of the dataset size. When the dataset size is small (5000) the number of ranges searches is relatively high. It even exceeds the total number of points. DBSCAN for each dataset requires at least n range searches to finalize, where n is the number of points in the dataset. Thus, for small datasets DEUC appears computationally expensive. On the other hand, as the dataset size increases, the efficiency of DEUC also increases. For example for 30000 points in the dataset DE_1 requires less than 6000 range searches, that is five times less than DBSCAN.

To demonstrate the quality of the partitioning results we employ two real world datasets. The first is the four dimensional Iris dataset $Dset_{iris}$ from the UCI Machine Learning Repository [9]. This dataset is among the best known databases to be found in the pattern recognition literature. It contains 150 records of four features. The features are measurements of the sepal and petal length and width of three

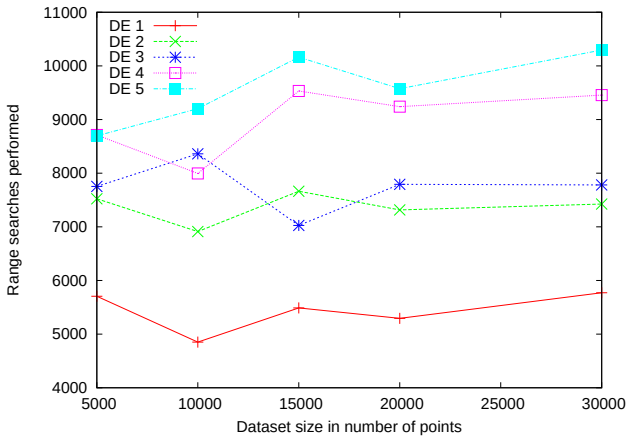

Figure 7: Mean number of range searches required for variable dataset size.

different types of the iris plant (Setosa, Versicolour and Virginica). The 150 records are equally distributed in three classes, each corresponding to a different type of the plant. To evaluate the clustering result we resolve to the correspondence they have to the true cluster labels of the patterns. Ideally, each cluster should contain patterns that belong to only one type of the Iris plant. After normalizing the data in the $[10, 100]^4$ range, DEUC was executed 100 times, using a population of 20 individuals while each individual encoded 5 d–ranges. In most cases 3 clusters were recognized by the algorithm, but there were also cases that resulted in 4 and 5 clusters. Moreover, as a comparison measure we executed DBSCAN using all the combinations of values in $[1, 10]$ with a step of 1, for the Eps and $MinPts$ parameters, yielding 100 different clustering results. In the box-plots exhibited in Fig. 8, we summarize the results with respect to the partitioning accuracy. Each box-plot depicts the obtained values for the classification accuracy, in the 100 experiments. The box has lines at the lower quartile, median, and upper quartile values. The lines extending from each end of the box (whiskers) exhibit the range covered by the remaining data. The outliers, i.e. the values that lie beyond the ends of the whiskers, are represented with crosses. Notches represent a robust estimate of the uncertainty about the median. As it is obvious from Fig. 8, all the different DE operators are able to capture the dynamics of the dataset and result in high partitioning accuracy. Among all the operators DE_3 exhibits the most robust behavior and is able to provide the best results even with respect to outliers. On the other hand, DBSCAN is unable to provide highly accurate results since in this dataset two of the classes are somewhat close and DBSCAN tends to merge them to a single cluster, thus destroying its classification accuracy.

The next dataset studied $Dset_L$ is the well–known and publicly available acute leukemia dataset provided by the center of genome research of the Whitehead Institute [16]. It is a well characterized dataset already used in numerous studies. This dataset contains mRNA expression profiles from 72 leukemia patients aiming at the development of an expression–based classification method for acute leukemia.

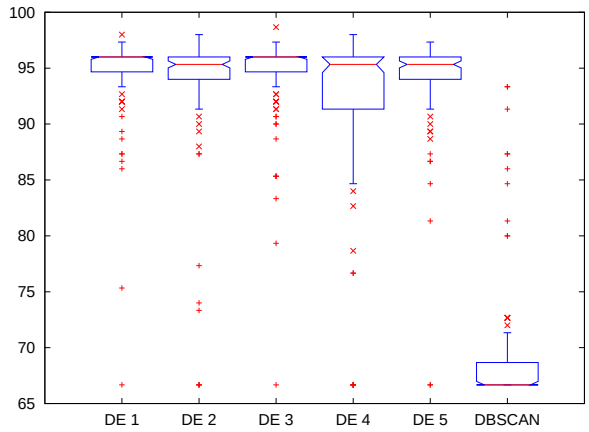

Figure 8: Box-plot of the classification accuracy for the Iris dataset

Each sample is measured over 7129 genes. The samples contain 47 acute myeloid leukemia (ALL) samples and 25 acute lymphoblastic leukemia (AML) samples. From the 7129 genes, the 50 most highly correlated genes with the ALL–AML class distinction, were selected as in [16]. Similarly, with the Iris dataset, all the values were normalized in the $[10, 100]^{50}$ range. Next 100 experiments of DEUC were performed and the partitioning accuracy of the clusters from each run with respect to the AML and ALL label of the patterns was recorded. The DBSCAN algorithm was also executed on the same dataset using all the combinations of values in $[40, 50]$ and $[1, 10]$ with a step of 1, for the Eps and $MinPts$ parameters respectively, yielding 100 different clustering results. The box-plots in Fig. 9, exhibit the obtained results. In this case, the DEUC algorithm reported 4-6 final clusters, with the classification accuracy obtaining values occassionaly exceeding 95%. In this case DE1 managed to perform better among the different variants of DE. The DBSCAN algorithm this time exhibited quite high classification results, similar to the DEUC algorithm. It should be noted, however, that it was very difficult to establish a good range for the Eps parameter due to the high dimensionality of the dataset (50).

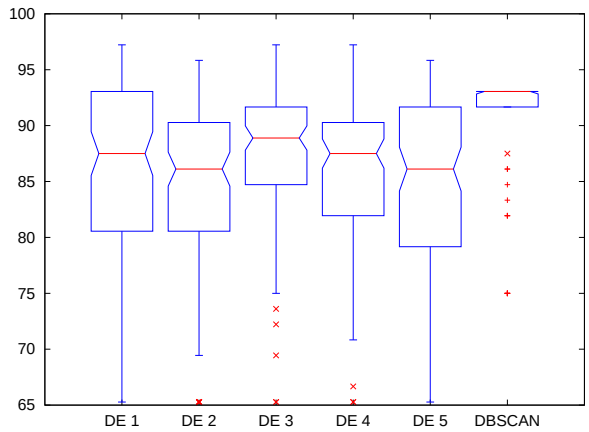

Figure 9: Box-plot of the classification accuracy for the Acute Leukemia dataset $Dset_L$

5 Concluding Remarks and Discussion

Many partitioning clustering algorithms based on evolutionary techniques have been proposed to tackle the problem of finding the optimal partition of a data set. Most of these approaches firstly encode a solution to the clustering problem as a chromosome, and then treat the clustering task as the optimization problem of locating the optimal centroids of clusters. The fitness of clustering solutions can be evaluated using various statistical functions [19, 33].

In the present contribution we present a new fitness function, that apart from being a meaningful clustering objective function, can be evaluated in sub-linear time with respect to the size of the dataset. This is achieved by utilizing Computational Geometry data structures.

Moreover, we propose an evolutionary scheme based on Differential Evolution that also has the ability to approximate the number of clusters and yields high quality partitions as it is evident from the experimental results provided. The proposed scheme can be applied using any evolutionary algorithm. In this paper we chose Differential Evolution, and aim to investigate the performance of different EAs in a future work. Furthermore, we wish to present an adaptive scheme, that will also be able to provide estimations for the parameter a of the WDF function.

Bibliography

[1] P. K. Agarwal and C. M. Procopiuc. Exact and approximation algorithms for clustering (extended abstract). In *9th Symposium on Discrete Algorithms*, pages 658–667, 1998.

[2] M. S. Aldenderfer and R. K. Blashfield. *Cluster Analysis*, volume 44 of *Quantitative Applications in the Social Sciences*. SAGE Publications, London, 1984.

[3] P. Alevizos. An algorithm for orthogonal range search in $d \geqslant 3$ dimensions. In *Proceedings of the 14th European Workshop on Computational Geometry*. Barcelona, 1998.

[4] G. P. Babu and M. N. Murty. A near optimal initial seed value selection in k-means algorithm using a genetic algorithm. *Pattern Recogn. Lett.*, 14(10):763–769, 1993.

[5] G. P. Babu and M. N. Murty. Clustering with evolution strategies. *Pattern Recogn.*, 27:321–329, 1994.

[6] R. W. Becker and G. V. Lago. A global optimization algorithm. In *Proceedings of the 8th Allerton Conference on Circuits and Systems Theory*, pages 3–12, 1970.

[7] J. L. Bentley and H. A. Maurer. Efficient worst-case data structures for range searching. *Acta Informatica*, 13:1551–68, 1980.

[8] J. N. Bhuyan, V. V. Raghavan, and K. E. Venkatesh. Genetic algorithm for clustering with an ordered rep-

resentation. In *Fourth International Conference on Genetic Algorithms*, page 408415, 1991.

[9] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[10] D. Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, 1998.

[11] B. Chazelle. Filtering search: A new approach to query-answering. *SIAM J. Comput*, 15(3):703–724, 1986.

[12] B. Chazelle and L. J. Guibas. Fractional cascading: Ii applications. *Algorithmica*, 1:163–191, 1986.

[13] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, Jan 2001.

[14] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. *Advances in Knowledge Discovery and Data Mining*. MIT Press, 1996.

[15] D. B. Fogel and P. K. Simpson. Evolving fuzzy clusters. In *International Conference on Neural Networks*, pages 1829–1834, 1993.

[16] T.R. Golub, D.K Slomin, P. Tamayo, C. Huard, M. Gaasenbeek, J. Mesirov, H. Coller, M.L. Loh, J. Downing, M. Caligiuri, C. Bloomfield, and E. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

[17] J. Handl and J. Knowles. Evolutionary multiobjective clustering. In *8th International Conference on Parallel Problem Solving from Nature (PPSN VIII) 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, pages 1081–1091, 2004.

[18] D. Jones and M. A. Beltramo. Solving partitioning problems with genetic algorithms. In *Fourth International Conference on Genetic Algorithms*, page 442449, 1991.

[19] F. H. C. Marriott. Optimisation methods of cluster analysis. *Biometrics*, 69(2):417422, 1982.

[20] N. Megiddo and K. J. Supowit. On the complexity of some common geometric problems. *SIAM Journal on Computing*, 13:182–196, 1984.

[21] S. Paterlini and T. Krink. Differential evolution and particle swarm optimization in partitional clustering. *Computational Statistics and Data Analysis*, 2005. To appear.

[22] F. Preparata and M. Shamos. *Computational Geometry*. Springer Verlag, New York, Berlin, 1985.

[23] V. Ramasubramanian and K. Paliwal. Fast k-dimensional tree algorithms for nearest neighbor search with application to vector quantization encoding. *IEEE Transactions on Signal Processing*, 40(3):518–531, 1992.

[24] M. Rigou, S. Sirmakessis, and A. Tsakalidis. A computational geometry approach to web personalization. In *IEEE International Conference on E-Commerce Technology (CEC'04)*, pages 377–380, San Diego, California, 2004.

[25] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu. Density-based clustering in spatial databases: The algorithm gdbscan and its applications. *Data Mining and Knowledge Discovery*, 2(2):169–194, 1998.

[26] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques, 2000. In KDD Workshop on Text Mining.

[27] R. Storn and K. Price. Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997.

[28] D. K. Tasoulis and M. N. Vrahatis. Unsupervised clustering on dynaic databases. *Pattern Recognition Letters*, 2005. to appear.

[29] A. Torn and A. Zilinskas. *Global Optimization*. Springer-Verlag, Berlin, 1989.

[30] C. Tryon. *Cluster Analysis*. Ann Arbor, MI: Edward Brothers, 1939.

[31] D. W. van der Merwe A. P. Engelbrecht. Data clustering using particle swarm optimization. In *Congress on Evolutionary Computation*, pages 215–220, Canberra, Australia, 2003.

[32] M. N. Vrahatis, B. Boutsinas, P. Alevizos, and G. Pavlides. The new k-windows algorithm for improving the k-means clustering algorithm. *Journal of Complexity*, 18:375–391, 2002.

[33] M.-S. Yang and K.-L. Wu. A similarity-based robust clustering method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(4):434–448, 2004.