EFFICIENT UNSUPERVISED CLUSTERING THROUGH INTELLIGENT OPTIMIZATION

G.S. Antzoulatos, E.K. Ikonomakis and M.N. Vrahatis
Computational Intelligence Laboratory (CILab)
Department of Mathematics
University of Patras, Greece
University of Patras Artificial Intelligence Research Center (UPAIRC)
email: antzoulatos@upatras.gr, {eki,vrahatis}@math.upatras.gr

ABSTRACT
A novel methodology for unsupervised data clustering based on Evolutionary Computation, named "Intelligent Unsupervised Clustering" (IUC) is introduced. IUC searches for the "optimal clusters' representatives" using Evolutionary Algorithms (EAs) and utilising a Window Density Function (WDF) as an objective function. EAs ensure that the representative is posed in a region of points of high density. IUC aims in finding a highly dense hyper-rectangle around the cluster's representative, that captures a part of cluster. Therefore, IUC uses a windowing technique and gradually enlarges a window, which is centered on the best individual generated from the EA. This process continues until the increase of the value of WDF does not change "drastically". The whole process is repeated on the unclustered data, until all the clusters are discovered. The quality of clustering, delivered by the IUC, is compared with well-known clustering algorithms and the experimental results illustrate its efficiency and accuracy.

KEY WORDS
Computational Intelligence, Evolutionary Computation, Data Clustering, Differential Evolution, Window Density Function

1. Introduction

Clustering is the process of identifying sets of similar items, called clusters. The goal of a clustering algorithm is to produce a set of clusters with high intra-cluster similarity while simultaneously preserving a low inter-cluster similarity [4, 13]. Its application domain consists of a broad collection of scientific fields including, statistics [2], bioinformatics [25], text mining [29], marketing and finance [5, 16, 22], image segmentation and computer vision [14] and pattern recognition [27], among others. Many approaches have been proposed, which can be categorised into two major categories, hierarchical and partitioning [4, 15].

Partitioning algorithms consider the clustering as an optimization problem. There are two directions. The first one discovers clusters through optimizing a goodness criterion based on the distance of the dataset's points. Such algorithms are k-means [17], ISODATA [3] and fuzzy c-means [6]. The second one utilizes the notion of density and considers clusters as high-density regions. The most characteristic algorithms of this approach are DBSCAN [10], CLARANS [18] and k-windows [28].

Recent approaches for clustering also apply evolution, which is an optimization process aiming in adjustment of an organism to survive in a dynamically changing and competitive environment. Evolutionary Computation (EC) refers to the computer-based methods that simulate the evolution process. Genetic algorithms (GA), Differential Evolution (DE) and Particle Swarm Optimization (PSO) are the main algorithms of EC [9]. The principal issues of these methods consist of the representation of the solution of the problem and the choice of the objective function.

The majority of clustering algorithms require a predefined number of clusters or at least an upper bound of them. This is a critical open issue in cluster analysis, since this information is often tough to determine or, even worse, impossible to define. Dubes called this problem "*the fundamental problem of cluster analysis*" [8].

In this study, a novel methodology is introduced, called "Intelligent Unsupervised Clustering" (IUC), utilising an Evolutionary Algorithm, which models the natural and biological intelligence, aiming to find a partitioning without any prior knowledge of the number of clusters present in the dataset (unsupervised clustering). Particularly, IUC exploits the benefits of EAs by deploying the Window Density Function [26] as an objective function, so as to discover the optimum clustering. The key idea is to discover the center of the most dense region of the dataset and then by using a windowing technique to capture the entire cluster. This process is repeated over the unclustered dataset. Scrutinising the effectiveness of the IUC algorithm, experimental runs have been done and the results were compared with other well known clustering algorithms, according to the entropy measure.

The rest of the paper is organised as follows. In Section 2 the partitional clustering algorithms that are used in this work are sketched. After that, the evolutionary clustering algorithms are shortly reviewed in Section 3. The proposed algorithm is outlined in Section 4 while in Section 5 the experimental results are demonstrated. Finally,

conclusions and outlines for future work are discussed in Section 6.

2. Partitional Clustering Algorithms

The most famous clustering algorithm is *k*-means [17], which minimizes the distances between the points and the center of the cluster that they belong to and maximizes the distances between these points from the centers of other clusters. This can be formalised by minimizing the square-error criterion, $\min \sum_{j=1}^{k} \sum_{x_i \in C_k} \|x_i - c_j\|^2$, where c_j is the mean (or weighted average) of the j-th cluster. The above criterion is based on the L_2-norm (Euclidean distance). The algorithm selects k initial centers either randomly or by exploiting any prior knowledge. The algorithm proceeds iteratively by reassigning each point to its nearest cluster center and recalculating the center of the clusters until an optimum value of the objective function is found. The *k*-means algorithm works well only for compact and hyper-spherical clusters and can not guarantee convergence to the global optimum.

Another well-known clustering algorithm is DBSCAN (Density Based Spatial Clustering of Applications with Noise) [10] that relies on a density-based notion of clusters and it can deal with arbitrarily shaped clusters in single-scan mode. DBSCAN aims to group adjacent points into clusters based on local density criterion. The neighbourhood of a given radius *eps* for each point of a cluster has to contain at least a minimum number of points *Input's*. An *eps*-neighbourhood is defined as the set $N_{eps}(p) = \{q \in X | d(p,q) \leqslant eps\}$. There are two categories of points in a cluster, core and border points. *MinPts* is a lower boundary of the points in an *eps*-neighborhood. So a core point of a cluster contains significantly more points than *MinPts* in its *eps*-neighbourhood (core point condition) and a boarder point contains at least *MinPts* points. Furthermore, a point p is *directly density reachable* from q with respect to *eps* and *MinPts* in the set of points X if the point p is in the *eps*-neighbourhood of q. A point p is *density reachable* from a point q with respect to *eps* and *MinPts* in the set of points X if there is a sequence of points $p_1, p_2, \ldots, p_n$, $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density reachable from p_i.

DBSCAN starts by randomly selecting a point of the database. If the core point condition is not fulfilled for this point, it is marked as noise and another point is visited. Else, all the density reachable points from the selected point are retrieved and clustered together. This process terminates when all points have been examined. The points that cannot be assigned to a cluster are considered as noise.

Vrahatis *et al.* proposed the *k*-windows algorithm [28] that uses a windowing technique to discover the underlying clusters. It starts spreading an initial number of d-dimensional rectangles, called windows, over the dataset. During the movement process each window moves iteratively so that its center approaches the mean of the points that lie in it. This iterative process is stopped when further movements do not significantly contribute to the increase in the number of points that lie in the window. Then, the current window is enlarged in order to capture as many patterns from the cluster as possible. For each coordinate, the window is augmented iteratively by a user-defined portion θ_ε until the increment of the points inside the window is insignificant. The movement and enlargement procedures are performed on all the windows. After that, the overlapping windows are considered for merging. During this operation, the number of points that lie in the intersection of each pair of overlapping windows is computed. If the ratio of this number to the number of patterns lying in each window exceeds a user-defined threshold then the smaller window is discarded, as the two windows are considered identical. If this ratio is low then these windows capture different clusters and therefore both of them are kept. Otherwise, the two windows capture different parts of the same cluster. The remaining windows are returned as the final clustering of the dataset.

3. Clustering using Differential Evolution

As mentioned above, Evolutionary Algorithms (EA) are bio-inspired optimization processes. The concept on which they are based is the evolution of possible solutions simulating the evolution of organisms. Storn and Price [23] proposed an EA algorithm called Differential Evolution (DE) that randomly initializes a population and evolves it over a number of generations employing a set of operators: mutation, recombination and selection.

More formally, consider the given function $f : A \to \mathbb{R}$, where $A \subseteq \mathbb{R}^d$, the minimization problem is to find the global minima x^* such that $f(x^*) \leqslant f(x)$ for $x \in A$. So on, f is called objective function. A population of candidate solutions NP, $x_{1,g}, x_{2,g}, \ldots, x_{NP,g}$ is formed, where g represents the generation of the population.

In this framework, DE starts by randomly initializing a population of individual $x_{i,0}$, $i = 1, 2, \ldots, NP$. Then mutation is applied by combining individuals randomly chosen and producing a new mutated individual using the following mutation operators [26]:

$$v_{i,g+1} = x_{best,g} + F \cdot (x_{r_1,g} - x_{r_2,g}) , \quad (1)$$

$$v_{i,g+1} = x_{r_1,g} + F \cdot (x_{r_2,g} - x_{r_3,g}) , \quad (2)$$

$$v_{i,g+1} = x_{i,g} + F \cdot (x_{best,g} - x_{i,g}) + F \cdot (x_{r_1,g} - x_{r_2,g}) , \quad (3)$$

$$v_{i,g+1} = x_{best,g} + F \cdot (x_{r_1,g} - x_{r_2,g}) + F \cdot (x_{r_3,g} - x_{r_4,g}) , \quad (4)$$

$$v_{i,g+1} = x_{r_1,g} + F \cdot (x_{r_2,g} - x_{r_3,g}) + F \cdot (x_{r_4,g} - x_{r_5,g}) . \quad (5)$$

where r_1, r_2, r_3, r_4, r_5 are randomly chosen integers from $\{1, 2, \ldots, NP\}$ and $F \in [0, 2]$ is a constant factor which controls the contribution of the difference between the individuals. In the rest of this paper, Eq. (1) will be referred as operator DE1, Eq. (2) as operator DE2 and so on.

During the next step of the algorithm (recombination), each mutant individual $v_{i,g+1}$ is combined with the

individual $x_{i,g}$ according to

$$u_{ji,g+1} = \begin{cases} v_{ji,g+1}, & \text{if } (rand(j) \leqslant CR) \text{ or } j = k\,, \\ x_{ji,g}, & \text{otherwise}\,, \end{cases} \quad (6)$$

where $i = 1,2,\ldots,NP$, $j = 1,2,\ldots,d$ and k is a randomly chosen parameter from $\{1,2,\ldots,d\}$ and $CR \in [0,1]$ is a user-defined constant. Finally, the trial individual $u_{i,g+1}$ is accepted for the next generation if and only if its fitness value is better than the initial individual $x_{i,g}$ (selection). The outcome of this process is to receive a new generation in which each individual is at least as good as the current one.

As clustering is regarded as an optimization problem the DE algorithm has already been applied for its solution. In the evolutionary clustering framework two main issues have emerged. The first one is the representation of the clustering as a member of the population evolved. The majority of the proposed schemes encodes the entire clustering via its cluster centers as a single individual [12, 19, 21, 26]. This means each chromosome encodes the k representatives as real-valued vector. However, Das *et al.* [7] integrated k activation thresholds in the individual. This encoding does not take for granted the presence of k clusters but sets an upper bound $(K_{\max})$ of the number of clusters. The activation threshold $T_{i,j} \in [0,1]$ represents whether the corresponding center $(\mathbf{m}_{i,j})$ is participating in the clustering described by the chromosome $\mathbf{x}_i$. A center $\mathbf{m}_{i,j}$ is active if its threshold exceeds 0.5.

The second issue that needs to be addressed, concerning DE Clustering, is the choice of the appropriate objective function so as to evaluate the clustering quality of each chromosome. In this direction, Paterlini and Krink [21] utilized statistical criteria as objective functions of GA, DE and PSO clustering and compared their performance. Gong *et al.* [12] proposed an evolutionary algorithm-based clustering, called density-sensitive evolutionary clustering (DSEC) by using a density-sensitive dissimilarity measure that can describe the distribution characteristic of data clustering. Recently, Das *et al.* [7] utilised two well-known clustering validity measures, namely CS and DB indices, as fitness functions. Finally, a variant approach for DE Clustering was introduced by Omran *et al.*, called Self-adaptive Differential Evolution (SDE) [19]. In their approach, the control parameters of DE process (F and CR that mentioned above) are defined by an automatic way. Also, they proposed a fitness function based on a quantization error measure.

All the aforementioned approaches exhibit high computational cost involved in the computation of the fitness function. In order to overcome this difficulty, Tasoulis and Vrahatis [26] introduced a novel objective function, called Window Density Function (WDF), that represents the number of points lying in a hyper-rectangle. They proposed an evolutionary scheme, called DEUC, that applies the DE algorithm so as to evolve the clustering solution. More specifically, each chromosome represents k predefined d-dimensional vectors, each of them being the center of a window. The objective function of a chromosome is the sum of the WDF function over all windows. The DE algorithm is executed iteratively until the majority of data is covered. Finally, DEUC adopts the merging process of k-windows so as to discover the real clusters.

4. Intelligent Unsupervised Clustering

The proposed methodology is based on the combination of an Evolutionary Algorithm with a windowing technique and aims to discover the clusters of the dataset one after the other. It should be clarified that, the proposed methodology can utilise any Evolutionary Algorithm, such as Genetic and Differential Evolution Algorithms, as well as any Computational Swarm Intelligence scheme such as PSO.

In this contribution, the Differential Evolution is applied to optimize the Window Density Function (WDF), in order to find a region of high density. After that, this region is continuously enlarged until a cluster or a part of it is captured. The patterns that reside in the cluster are removed from the dataset and the process is repeated on the remaining points to detect another cluster. This process is terminated when only noise or outliers are left. The final number of clusters is produced by merging the overlapping windows. Furthermore, it must be noted that each chromosome is represented by the center of a single window and thus the dimensionality of the search space for the DE algorithm is not increased. Before the detailed description of the proposed algorithm is exhibited, WDF is formally defined.

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

$$S_{\alpha,z} = \{y \in X : z_i - \alpha \leqslant y_i \leqslant z_i + \alpha,\ \forall\, i = 1,2,\ldots,d\}\,.$$

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

$$WDF_{\alpha}(z) = |S_{\alpha,z}|\,. \quad (7)$$

WDF is a meaningful non-negative function that expresses the density of the region (orthogonal range) around the point. The points that are included in this region can be effectively estimated using computational geometry methods [1]. For a given α, the value of WDF increases continuously as the density of the region within the window increases. Furthermore, for low values of α, WDF has many local maxima, more than the real number of clusters. While the value of α increases, WDF reveals the number of local maxima that corresponds to the number of clusters. However for higher values of the parameter, WDF becomes smoother and the clusters are not distinguished (Fig. 1).

The discovery of high density regions of the datasets through the WDF is a maximization problem, however DE is a minimization algorithm, hence $-WDF$ is utilised as

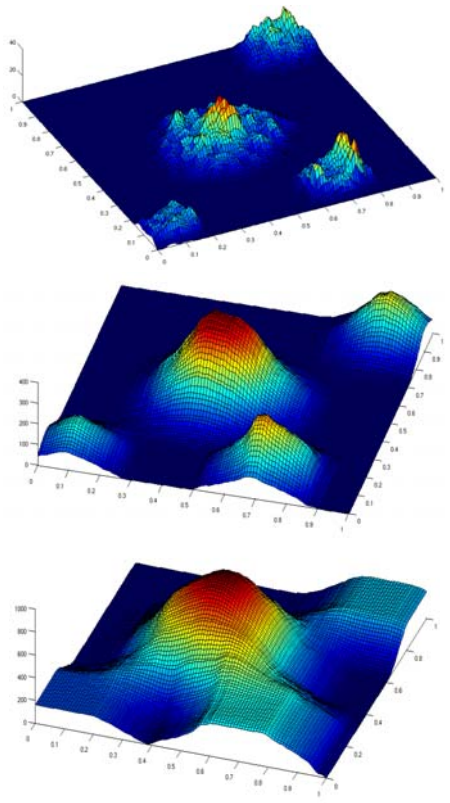

Figure 1. WDF plot for $a = 0.02$, $a = 0.1$ and $a = 0.2$

Algorithm 1 Intelligent Unsupervised Clustering - IUC

repeat

Create a data structure that holds all unclustered points

Perform the DE algorithm returning the center z

Construct the window w of size α around the center z

Enlarge the window w for each coordinate until

$$\frac{|Q_{w'}| - |Q_w|}{|Q_w|} \leqslant v \tag{8}$$

is satisfied

Mark the points that lie in the enlarged window w' as clustered

Remove the clustered points from the dataset

until

$$WDF_a(z) \leqslant WDF_a(best) \cdot t\% \tag{9}$$

Mark the points that lie in overlapping windows as members of the same cluster and merge these windows to form the clusters.

the fitness function. IUC uses $-WDF$ as shown in Algorithm 1.

Analytically, during the first step IUC constructs a data structure in which the unclustered data are stored. Then, the DE algorithm is executed in order to detect the point z with the property that the window, which is centered to it, has the highest density with respect to WDF function and a given parameter α. Due to the fact that DE usually converges to the global optimum, the window that encloses this optimum is the region with the highest density. As it has been mentioned before the user-defined parameter α influences the convergence of DE algorithm, in the sense that as its value decreases, the number of dense regions is increased. Thus, the DE algorithm is prone to converge to a local optima, which corresponds to a region with lower density. However, the detection of such a cluster prior to one of higher density does not affect the performance of IUC. On the other hand, large values of α produce overlapping regions whilst they span the search space covering more that one cluster.

In the third step, a window w of size α centered at z is constructed. This window represents the core of the cluster. During the fourth step, this window w is enlarged as much as possible over one coordinate and, afterwards the next coordinate is considered. Let, w denote the current window, w' the enlarged window, and Q_w, $Q_{w'}$ the set of points lying in windows w, w'. A window w is resized over each coordinate by a user-defined percentage e, as long as the enlargement termination criterion, inequality (8), does not hold. Parameter v is user-defined and represents the upper bound of the relative difference of the cardinality of the windows. It means that, when the increment of points is larger than v% of the number of points in the initial window, $|Q_w|$, then the enlargement process is continued. Otherwise, the enlargement procedure proceeds to the next coordinate. This step terminates when all the coordinates have been considered. Choosing large values of enlargement parameter e causes a large increasement of the window's size attaining quick coverage of the region, but endangering to capture contiguous regions. Given a constant value for the parameters α and e, parameter v represents the sensitivity of the algorithm to capture a dense region in a window. Small values of v allow IUC to enlarge a window covering bigger portion of the region. On the other hand, bigger values of v deter IUC for excessive enlargements.

The points inside the final window belong to the same cluster and so they are marked with a cluster id. IUC removes them from the dataset and repeats the previous steps on the new dataset, in order to capture the next cluster. This is done until the WDF value of DE's best individual in the current run falls under t% of the largest encountered WDF value by DE algorithm. The parameter t is a threshold of what IUC considers a dense region in the sense that t is a boundary of the expansion of a density region over a sparse one. Finally, two windows are merged if they overlap. Due to the fact that a region of high density may have an arbitrary shape that can not be enclosed in a single hyper-rectangle, such regions are described by several windows which partially overlap. Therefore, these windows must be merged in order to capture an entire cluster. A common cluster id is assigned to the points that lie in them. This step yields the final number of clusters in the dataset, while

the remaining points are considered as noise or possibly as outliers.

It must be stressed that IUC clusters a dataset in an unsupervised manner, since it detects the clusters without a priori knowledge of their number. It is based solely on the density of a region. Although for the execution of IUC algorithm, a user must determine the parameters α, e, v and t, these user-defined parameters are easily regulated, in contrast with the number of clusters that is an invariant feature characterising the underlying structure of the dataset and furthermore it is difficult to define. Also, DE's search space dimension is equivalent to the dimensionality of the dataset, in contrast to the majority of other approaches that try to find all centers simultaneously, increasing the dimensionality by a factor of k (where k denotes the maximum number of clusters estimated).

5. Experimental Results

In this section, the IUC algorithm is compared to four partitioning clustering algorithms, k-means, DBSCAN, k-windows and DEUC. All algorithms are implemented using the C++ programming language on the Linux operating system. For each dataset, the algorithms are executed 100 times, except DBSCAN that due to its deterministic nature was executed once. For the k-means algorithm the parameter k is set equal to the real number of clusters in each dataset. For the other algorithms, their parameters were determined heuristically. Finally, for the algorithms DEUC and IUC all the mutation operators (Eqs. (1)-(5)) are utilized in order to investigate their effects on clustering quality.

Furthermore, for the evaluation of IUC's performance, two 2-dimensional, one 3-dimensional and one 5-dimensional artificial datasets (called $Dset_1$, $Dset_2$, $Dset_3$ and $Dset_4$ respectively) are used, as shown in Fig. 2. The first one, $Dset_1$ has 1600 points that form four spherical clusters each one with different size. The second one, $Dset_2$ has 2761 points grouping into four arbitrary shape clusters, three of them are convex and one is non-convex. The remaining datasets, $Dset_3$ and $Dset_4$, contain 15000 points each and randomly generated from multivariate normal distributions, the first of them with an unary covariance matrix and different mean vectors, grouping into six clusters with different cardinality, based on [21] and the latter with random parameters, forming eight clusters, based on [24]. All the datasets are normalized in the $[0,1]^d$ range.

It is well known that Evolutionary Algorithms require more execution time than classical clustering algorithms [15, 20], thus this work is focused on the clustering quality rather than the execution time. In order to evaluate the performance of the clustering algorithms the Entropy and Purity measures are utilised. The Entropy function [29] represents the dissimilarity of the points lying in a cluster. Higher homogeneity means that entropy's values converge to zero. However, for the usage of the entropy function, the knowledge of the real classification/categorization

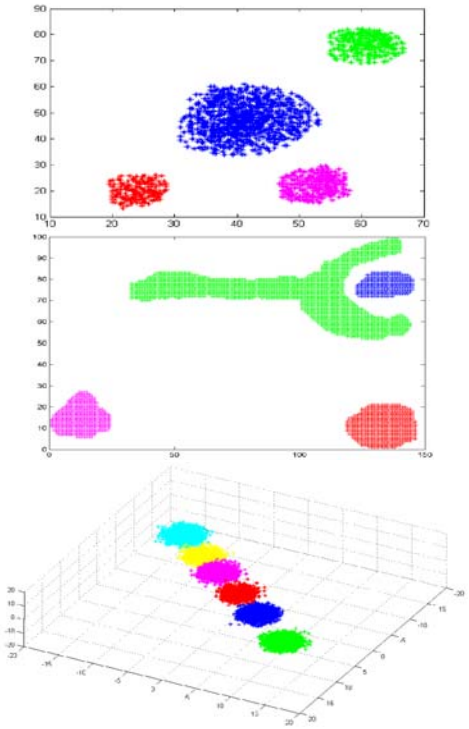

Figure 2. Datasets $Dset_1$, $Dset_2$ and $Dset_3$

of the points is required. Let, $\mathscr{C} = \{C_1, C_2, \ldots, C_k\}$ be a clustering provided by a clustering algorithm and $\mathscr{L} = \{L_1, L_2, \ldots, L_m\}$ be the target classification of the patterns, then the entropy of each cluster C_i is defined as $H_i = -\sum_{j=1}^{m} P(x \in L_j | x \in C_i) \log P(x \in L_j | x \in C_i)$. For a given set of n patterns, the entropy of the entire clustering is the weighted average of the entropy of each cluster. The Purity is defined as $r = \frac{1}{n} \sum_{i=1}^{k} \alpha_i$, where k denotes the number of clusters found in the dataset and α_i represents the number of patterns of the class to which the majority of points in cluster i belongs to it [11].

The experimental results for the dataset $Dset_1$ show that the proposed algorithm (IUC) attains to distinguish the four clusters in the majority of the experiments, as the average entropy tends to be zero and the average purity tends to 100%. The IUC algorithm outperforms both the DEUC and k-means algorithms, independently of the choice of mutation operator, whilst the average entropy and purity of the former algorithm is better than the average measure values of the latter algorithms. Furthermore, IUC's performance is equivalent to the other classical partitioning algorithms (Table 1). In contrast to the DEUC algorithm, IUC exhibits the potential to cluster correctly all the points of the dataset, as k-means, DBSCAN and k-windows achieved, as presented in Table 2. The parameters for the IUC algorithm were set to $\alpha = 0.1$, $e = 0.15$, $v = 0.05$ and $t = 0.001$.

According to the experimental results for the second dataset, some interesting conclusions are reached. IUC's ability to cluster all the points in the dataset remains as in the precedent dataset (Table 2). However, this fact has

a slight impact to its performance as its average entropy is higher than the average entropy of both the DEUC and the *k*-windows algorithms. On the other hand, IUC's mean entropy value for each strategy is significantly lower than the DBSCAN's and *k*-means' performance. In addition, IUC's purity outperforms the purity of DEUC and also *k*-means and *k*-windows but does not surpass DBSCAN's purity (Table 1). All the algorithms, except *k*-windows, encouter difficulties to distinguish the exact number of clusters, due to presence of the non-linear separable clusters in the dataset. Specifically, IUC splitted the dataset into 10 to 15 clusters in the majority of the experiments, while DEUC and DBSCAN found on the average 38 and 7 clusters respectively. The IUC's experiments were performed for $\alpha = 0.03,\ e = 0.02,\ v = 0.01$ and $t = 0.01$.

In the third dataset, IUC's performance is almost equivalent to DEUC's and *k*-means performance regarding the entropy, but significantly better regarding the purity and the points that it attained to cluster. Compared to DBSCAN and *k*-windows, IUC still lacks in performance, regardless of the choice of the utilised measure (Table 1). Although, IUC could not cluster every point, it could manage to cluster a higher percentage of points than the other DE clustering algorithm (Table 2). IUC and *k*-windows found the actual number of clusters in contrast with DEUC and DBSCAN that group data in more than 6 clusters. The best values for the IUC's parameters resulting for preliminary experiments were $\alpha = 0.09,\ e = 0.05,\ v = 0.05$ and $t = 0.01$.

Finally, while the dimensionality and the number of clusters increases, the entropy and purity values of IUC are considerably better than the corresponding values of DEUC and *k*-means, as shown in the results for the 5-dimensional dataset $Dset_4$ (Table 1). On the other hand, IUC's performance is slightly worse than *k*-windows' and DBSCAN's. DEUC could not manage to cluster as many points as IUC and the classical partitioning algorithms did. As regarding the number of clusters that algorithms could detect, IUC and *k*-windows manage to discover 8 clusters in contrast with other algorithms that found on average 7 clusters. The IUC's experiments were performed for $\alpha = 0.2,\ e = 0.05,\ v = 0.01$ and $t = 0.001$.

Generally speaking, the above conclusions can be confirmed by using statistical hypothesis tests such as the Kruskal-Wallis test. For every dataset Kruskal-Wallis tests were executed and resulted in near zero p-values, which means that the null hypothesis is rejected, leading to the conclusion that the mean entropy and purity of the algorithms are different.

With regard to the execution time of the proposed algorithm, it can be concluded that for the 2-dimensional dataset, IUC is outperformed by the other algorithms, but as the dimensionality and the cardinality of the dataset increase, the IUC algorithm exhibits better behaviour. More specifically, it is significantly faster than DEUC, however is slightly slower than traditional partitioning algorithms. Although, this is an encouraging result, extended research needs to be done on the running time and complexity issues of the algorithm, that is beyond of the scope of this work.

6. Conclusion

In this work, a novel methodology is proposed based on Evolutionary Computation so as to detect the clusters of a dataset in an intelligent and unsupervised manner. In more detail, a Differential Evolution algorithm is utilized in combination with an efficient and simple objective function, called Window Density Function, in order to discover the centers of regions of high density successively. After the enlargement of this region the final window will arise and the points in it are marked and removed. The repetition of this process produces the final windows and after the merging procedure the clusters of the dataset are produced without any prior knowledge about their number. Furthermore, the proposed codification of the individual is more efficient than contemporary encodings, as each individual represents a cluster, thus avoiding the increase of the dimensionality.

The experimental results show that the IUC algorithm is a promising algorithm while its performance is better than the other DE clustering algorithm (DEUC) and also is at least comparable with other well known partitioning algorithms. In addition, it exhibits good behaviour regarding the scalability, thus its performance is not affected when the dimensionality as well as the size of the dataset and the number of clusters increase. Although, more work needs to be done to discover the intrinsic behaviour of the IUC algorithm over various multidimensional datasets with arbitrary shape and different densities. Furthermore, in a future correspondence the time complexity of IUC needs to be considered. Also, it is interesting to investigate the proposed methodology using swarm computing (PSO) instead of the DE algorithm. Finally, the behaviour of the IUC algorithm needs to be examined when treating datasets with noisy data and/or outliers.

References

[1] P. Alevizos, B. Boutsinas, D. K. Tasoulis, and M. N. Vrahatis. Improving the orthogonal range search *k*-windows algorithms. In *14th IEEE International Conference on Tools and Artificial Intelligence*, pages 239–245, 2002.

[2] P. Arabie and L. J. Hubert. *Clustering and Classification*, chapter An overview of combinatorial data analysis, pages 5–64. World Scientific Publishing Co, 1996.

[3] G. Ball and D. Hall. A clustering technique for summarizing multivariate data. *Behavioral Sciences*, 12:153–155, 1967.

[4] P. Berkhin. Survey of data mining techniqes. Technical report, Accrue Software, 2002.

[5] M. J. A. Berry and G. Linoff. *Data mining techniques for marketing, sales and customer support*. John Willey & Sons Inc., USA, 1996.

[6] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, USA, 1981.

[7] S. Das, A. Abraham, and A. Konar. Automatic clustering using an improved differential evolution algorithm. *IEEE Transactions on Systems, Man and Cybernetics*, 38:218–237, 2008.

[8] R. Dubes. *Handbook of Pattern Recognition and Computer Vision*, chapter Cluster Analysis and Related Issue, pages 3–32. World Scientific, Singapore, 1993.

[9] A. P. Engelbrecht. *Computational Intelligence: An Introduction*. John Wiley & Sons, Ltd., 2007.

[10] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[11] J. Ghosh and A. Strehl. *Similarity-based Text Clustering: A Comparative Study*, chapter Criterion Functions for Clustering on High-Dimensional Data, pages 73–97. Springer-Verlag, Berlin Heidelberg, 2006.

[12] M. Gong, L. Jiao, L. Wang, and L. Bo. Density-sensitive evolutionary clustering. In *11th Pacific-Asia Conference in Advances in Knowledge Discovery and Data Mining*, volume 38, pages 507–514, 2007.

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

[14] A. K. Jain and P. J. Flynn. *Advances in Image Understanding: A Festschrift for Azriel Rosenfeld*, chapter Image segmentation using clustering, pages 65–83. Willey - IEEE Computer Society Press, Singapore, 1996.

[15] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31:264–323, 1999.

[16] F. Lisi and M. Corazza. *Mathematical and Statistical Methods in Insurance and Finance*, chapter Clustering financial data for mutual fund managment, pages 157–164. Springer, Milan, 2007.

[17] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[18] R. Ng and J. Han. CLARANS: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, 14(5):1003–1016, 2002.

[19] M. G. H. Omran and A. P. Engelbrecht. Self-adaptive differential evolution methods for unsupervised image classification. In *Proceedings of IEEE Conference on Cybernetics and Intelligent Systems*, pages 1–6, 2006.

[20] M. G. H. Omran, A. P. Engelbrecht, and A. A. Salman. An overview of clustering methods. *Intelligent Data Analysis*, 11(6):583–605, 2007.

[21] S. Paterlini and T. Krink. Differential evolution and particle swarm optimisation in partitional clustering. *Computational Statistics & Data Analysis*, 50:1220–1247, 2006.

[22] N. G. Pavlidis, V. P. Plagianakos, D. K. Tasoulis, and M. N. Vrahatis. Financial forecasting through unsupervised clustering and neural networks. *Operations Research - An International Journal*, 6(2):103–127, 2006.

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

[24] D. K. Tasoulis. http://stats.ma.ic.ac.uk/d/dtasouli/public_html.

[25] D. K. Tasoulis, V. P. Plagianakos, and M. N. Vrahatis. Unsupervised clustering in mRNA expresion profiles. *Computers in Biology and Medicine*, 36:1126–1142, 2006.

[26] D. K. Tasoulis and M. N. Vrahatis. The new density function for efficient evolutionary unsupervised clustering. In *IEEE Congress on Evolutionary Computation, CEC 2005*, volume 3, pages 2388–2394. IEEE Press, 2005.

[27] S. Theodoridis and K. Koutroubas. *Pattern Recognition*. Academic Press, 1999.

[28] 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.

[29] Y. Zhao and G. Karypis. *Grouping Multidimensional Data Recent Advances in Clustering*, chapter Criterion Functions for Clustering on High-Dimensional Data, pages 211–237. Springer-Verlag, Berlin Heidelberg, 2006.

Table 1. The mean values and standard deviation of entropy and purity for each algorithm over the four datasets.

Entropy Values								
	$Dset_1$		$Dset_2$		$Dset_3$		$Dset_4$	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
IUC DE1	8.55e-3	6.08e-2	4.54e-2	1.14e-1	2.52e-3	2.52e-2	2.7e-3	2.65e-2
IUC DE2	1.80e-2	1.03e-1	3.08e-2	9.12e-2	7.59e-3	4.34e-2	7.9e-3	4.54e-2
IUC DE3	*1.94e-4*	1.94e-3	7.16e-2	1.34e-1	1.02e-2	5.00e-2	8.0e-3	4.54e-2
IUC DE4	6.01e-3	6.01e-2	4.21e-2	1.02e-1	**0.00e+0**	0.00e+0	1.06e-3	5.22e-2
IUC DE5	2.46e-2	1.21e-1	6.95e-2	1.33e-1	5.04e-3	3.55e-2	2.12e-3	7.22e-2
DEUC DE1	1.70e-1	1.02e-1	3.39e-2	1.87e-2	6.86e-3	8.92e-3	2.63e-3	2.14e-1
DEUC DE2	1.36e-1	9.90e-2	3.22e-2	1.73e-2	6.04e-3	8.73e-3	2.90e-3	1.93e-1
DEUC DE3	1.66e-1	9.35e-2	2.90e-2	1.86e-2	6.16e-3	7.43e-3	2.94e-3	2.14e-1
DEUC DE4	1.45e-1	9.20e-2	3.16e-2	1.93e-2	7.17e-3	1.15e-2	3.09e-3	2.4e-1
DEUC DE5	1.39e-1	1.04e-1	*2.88e-2*	1.81e-2	6.38e-3	8.84e-3	2.79e-3	2.29e-1
k-means	1.10e-1	2.15e-1	3.45e-1	5.70e-2	2.69e-1	1.89e-1	3.99e-3	2.55e-1
k-windows	**0.00e+0**	0.00e+0	**2.20e-2**	8.14e-2	4.18e-5	3.06e-4	0	0
DBSCAN	**0.00e+0**	—	3.74e-1	—	8.54e-4	—	0	0
Purity Values								
	$Dset_1$		$Dset_2$		$Dset_3$		$Dset_4$	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
IUC DE1	99.7%	2.16%	98.9%	2.7%	94.7%	4.75%	*99.7%*	1.01%
IUC DE2	99.4%	3.29%	*99.2%*	2.26%	96%	4.26%	99.5%	1.6%
IUC DE3	**100%**	0.02%	98.2%	3.34%	95.5%	4.01%	99.6%	1.6%
IUC DE4	99.8%	1.97%	99%	2.53%	96.6%	1.65%	99.4%	1.83%
IUC DE5	99.2%	3.88%	98.3%	3.28%	*97%*	1.78%	99%	2.55%
DEUC DE1	91.0%	5.22%	90.5%	1.35%	90.7%	2.04%	87.4%	7.11%
DEUC DE2	92.3%	5%	90.3%	1.11%	91%	2.09%	86.4%	5.76%
DEUC DE3	90.4%	5.17%	90.8%	1.23%	91.2%	1.64%	86.4%	7.09%
DEUC DE4	91.1%	4.63%	90.4%	1.35%	89.9%	1.88%	86%	7.51%
DEUC DE5	92.9%	5.19%	90.6%	1.08%	90.1%	2.11%	86.8%	7.04%
k-means	96.7%	6.5%	90.5%	3.35%	89.9%	7.15%	86.8%	9.01%
k-windows	99.2%	2.9%	95.4%	1.95%	98.3%	0.32%	99.7%	0.57%
DBSCAN	**100%**	—	**100%**	—	**99.2%**	—	**100%**	—

Table 2. The mean values and standard deviation of clustered points for each algorithm over the four datasets.

Clustered Points								
	$Dset_1$		$Dset_2$		$Dset_3$		$Dset_4$	
	Mean	STD	Mean	STD	Mean	STD	Mean	STD
IUC DE1	**1600.0 (100.0%)**	0.0	**2761.0 (100.0%)**	0.0	14220.5 (94.8%)	705.7	14961.7 (99.7%)	60.0
IUC DE2	**1600.0 (100.0%)**	0.0	**2761.0 (100.0%)**	0.0	14447.0 (96.3%)	596.7	14971.9 (99.8%)	13.0
IUC DE3	**1600.0 (100.0%)**	0.0	**2761.0 (100.0%)**	0.0	14383.9 (95.9%)	551.3	*14977.7 (99.9%)*	25.0
IUC DE4	**1600.0 (100.0%)**	0.0	**2761.0 (100.0%)**	0.0	14484.2 (96.6%)	247.7	14971.1 (99.8%)	18.5
IUC DE5	**1600.0 (100.0%)**	0.0	**2761.0 (100.0%)**	0.0	*14575.8 (97.2%)*	176.6	14953.4 (99.7%)	24.2
DEUC DE1	1540.2 (96.3%)	52.6	2524.0 (91.4%)	32.7	13633.3 (90.9%)	300.9	14332.6 (95.6%)	394.7
DEUC DE2	1545.9 (96.6%)	47.2	2516.5 (91.2%)	27.5	13671.7 (91.1%)	303.3	14281.2 (95.2%)	402
DEUC DE3	1527.6 (95.5%)	51.6	2527.2 (91.5%)	30.9	13703.0 (91.3%)	243.5	14336.6 (95.6%)	381.9
DEUC DE4	1532.4 (95.8%)	49.5	2520.0 (91.3%)	30.4	13516.2 (90.0%)	291.6	14306.9 (95.4%)	406.8
DEUC DE5	1541.7 (96.4%)	54.1	2522.8 (91.4%)	27.8	13535.0 (90.0%)	313.8	14303.4 (95.4%)	426.1
k-means	**1600.0 (100.0%)**	0.0	**2761.0 (100.0%)**	0.0	**15000.0 (100.0%)**	0.0	**15000 (100%)**	0
k-windows	1587.1 (99.2%)	46.4	2646.0 (95.8%)	20.4	14755.01 (98.4%)	47.6	14958.1 (99.7%)	86
DBSCAN	1587 (99.2%)	—	**2761 (100.0%)**	—	14887 (99.2%)	—	**15000 (100%)**	—