Pattern Recognition 87 (2019) 190–202

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Content driven clustering algorithm combining density and distance functions

Emmanouil K. Ikonomakis[a,*], George M. Spyrou[b,1], Michael N. Vrahatis[a]

[a] Computational Intelligence Laboratory (CILab), Department of Mathematics, University of Patras, Patras GR-26110, Greece
[b] Bioinformatics ERA Chair, The Cyprus Institute of Neurology and Genetics, 6 International Airport Avenue, Ayios Dometios, Nicosia 2370, Cyprus

ARTICLE INFO

Article history:
Received 12 January 2017
Revised 23 August 2018
Accepted 8 October 2018
Available online 12 October 2018

Keywords:
Clustering algorithms
Density based clustering
Distance based clustering
Evolutionary clustering
Window density function

ABSTRACT

Density and distance based clustering are two distinct approaches to the same problem. In this contribution, a novel algorithm is presented in order to exploit the benefits of both approaches. This is achieved, not by combining those approaches into a single notion, but by utilizing the advantages of each one, depending on what each step of the algorithm aims to achieve. To be precise, the Window Density Function is utilized to provide regions of high density and hence a region of clusters or a part of a cluster. Affinity Propagation is, consequently, utilized to provide a group of clusters within such a region. Finally, these regions are merged to form actual clusters. The proposed methodology is tested on a variety of synthetic and real-life datasets. The algorithm presented in this contribution outperforms other well-known algorithms, with which it is compared to, in the majority of the datasets used.

1. Introduction

Clustering is the process of identifying groups (clusters) of objects within a dataset. There are several approaches to solve this problem and many of them can be classified depending on their angle on measuring the proximity of the objects in a cluster. Furthermore, a clustering algorithm can be described as density-based if it operates based on the density of a region of the dataset or as similarity-based if it is based on the similarities among the members of a dataset [1–7]. On the other hand, algorithms can be classified based on their approach of detecting such clusters. For example, hierarchical algorithms [8–11] form a tree-like structure describing the dataset. Alternatively, partitioning algorithms provide clusterings of the dataset [12–18]. Moreover, new algorithms have emerged utilizing intelligent optimization methods [19–24]. As clustering can be approached as an optimization problem, algorithms such as Differential Evolution (DE) [25], Particle Swarm Optimization (PSO) [26] and Genetic Algorithms (GA) [27,28] have been used in order to tackle this problem. Nevertheless, most of these approaches attempt to cluster a dataset by trying to, simultaneously, identify the position of all cluster centers. Such an approach is only possible when the number of clusters can be determined in advance.

The introduced algorithm, Density and Distance Content-driven(DeDiCo) Clustering Algorithm has the ability to, automatically, determine the number of clusters without any assumptions. Additionally, it can determine whether it should use an evolutionary optimization algorithm or not. This decision is based upon the number of density evaluations that are expected when using such an algorithm. Therefore, it can adapt its behavior in order to reduce its running time. Additionally, given a d dimensional dataset with k clusters, the evolutionary optimization algorithm used would perform searches in a d dimensional space, instead of the $k \cdot d$ dimensional space, as the vast majority of evolutionary clustering algorithms. The algorithm presented in this work aims to combine distance with density metrics and therefore provide a new hybrid approach blending two clustering approaches. Yet, instead of forming a merging of both notions, distance and density are used separately. Specifically, each notion is used depending on the goal of each step of the algorithm. A density based approach can determine more easily regions of high density and hence identify cluster cores. On the other hand, a distance based algorithm would require several distance evaluations that may lead to a significant computational cost. Concerning the identification of the closest cluster, a density approach would not be able to address this issue as efficiently as a distance based approach.

The algorithm presented in this contribution utilizes the Window Density Function (WDF) [4,15] and a similarity-based ap-

* Corresponding author.
E-mail addresses: eki@math.upatras.gr (E.K. Ikonomakis), georges@cing.ac.cy (G.M. Spyrou), vrahatis@math.upatras.gr (M.N. Vrahatis).

[1] George M. Spyrou holds the Bioinformatics ERA Chair Position funded by the European Commission Research Executive Agency (REA) Grant BIORISE (Num. 669026), under the Spreading Excellence, Widening Participation, Science with and for Society Framework.

https://doi.org/10.1016/j.patcog.2018.10.007
0031-3203/

proach, Affinity Propagation (AP) [29]. WDF has already been introduced by Tasoulis and Vrahatis [4]. Yet, they identify all clusters simultaneously, as it is accustomed for most cases in Evolutionary Clustering. As mentioned earlier, this increases dramatically the search space of the evolutionary algorithm used. In attempt to address this, Antzoulatos et al. [15] introduced an algorithm named IUC that iteratively detects the clusters in a dataset. Although this resolves the problem of the dimensionality of the search space, IUC utilized evolutionary optimization even if it were computationally inefficient. Moreover, in the case of arbitrary shaped clusters, IUC had to be tuned in a way that resulted in lengthy runs.

On the other hand, AP is very efficient in addressing arbitrary shaped clusters. Unfortunately, AP needs to store three values for each pair of data points. For a dataset of n points, this translates to $3 \cdot n^2$ values, which can have serious implications regarding memory management.

In short, DeDiCo utilizes WDF to provide an initial clustering and then relies on AP to crystallize the clusters at hand. DeDiCo uses WDF in order to locate areas of high density, as those are considered to be the clusters cores. Yet, it does so, by determining the computationally more efficient approach. The next step aims at refining the results of the algorithm's first step. This is achieved by moving and enlarging the estimated cluster core in an atempt to better describe a region of higher density. Yet, such a region could easily incorporate parts of two or more clusters, decreasing the performance of the algorithm. Therefore, a thorough clustering of the points inside this region is deemed necessary and AP is applied on this region. By applying AP only on fractions of the dataset, where it is deemed to be useful, its complexity is reduced, while its ability to identify arbitrary shaped clusters is exploited. An undesired effect of this approach is to partition the dataset into several small clusters. Although, this allows the separation of the adjacent clusters and it also leads to segmenting clusters. Therefore, a final step of merging the provisional clusters of close proximity into a larger cluster provides a correction of this clustering approach. Finally, for performance purposes, some points may be left unclustered by the algorithm. If no outliers are expected, the user may choose to merge those points into the closest cluster.

In general, it can be said that DeDiCo defines high density regions in which AP is applied to refine the clustering. The distance attribute of the algorithm is justified as AP is utilized with the Euclidean distance. The proposed method is described in Section 2. In Section 3 the datasets used and the methods of experimental evaluation are presented. DeDiCo is compared to other well known algorithms, on both synthetic as well as real datasets. Section 4 presents the results on those datasets as well a deeper insight to the results of the experiments. Finally, Section 5 concludes this paper by summarizing the contributions of this work.

2. Density and Distance Content-driven (DeDiCo) clustering algorithm

In this contribution, the notion of density is combined with the notion of distance as parts of an algorithm, which swaps between the two notions instead of combining these two notions together. Its main idea is, simply, to identify regions of high density and to apply distance based clustering to these areas. Additionally, it merges clusters together by the notion of distance. More specifically, *Density and Distance Content-driven* (DeDiCo) Clustering Algorithm aims at determining regions of high density within the dataset. These regions are the base of a cluster or a cluster part. Around these clusters a hyper-rectangle will be formed and moved as well as enlarged so that a cluster is captured as accurately as possible. AP is applied to the points inside this hyper-rectangle and final cluster parts are formed. The points clustered with AP are removed from the dataset and the process is repeated until no more points are left.

2.1. Brief introduction to the notions used by DeDiCo

Before proceeding with a more thorough description of DeDiCo, for completeness purposes the Window Density Function (WDF) [4,15] is briefly described.

Definition 1 (Window)**.** Let **window** $w_\alpha(z)$ be a d- dimensional orthogonal range of size $\alpha \in \mathbb{R}$:

$$w_\alpha(z) = \left[z_1 - \frac{\alpha}{2}, z_1 + \frac{\alpha}{2}\right] \times \cdots \times \left[z_d - \frac{\alpha}{2}, z_d + \frac{\alpha}{2}\right].$$

Point $z = [z_1, z_2, \ldots, z_d]^\top$ is called the **center** of $w_\alpha(z)$. Additionally:

Definition 2. Given a dataset $S \subset \mathbb{R}^d$, the set $S_{\alpha,\, z}$ is defined as

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

Practically, $S_{\alpha,\, z}$ is the set of all points of S inside the window $w_\alpha(z)$.

Definition 3 (WDF)**.** For the set S with respect to a given α, the **Window Density Function** (*WDF*) is defined as

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

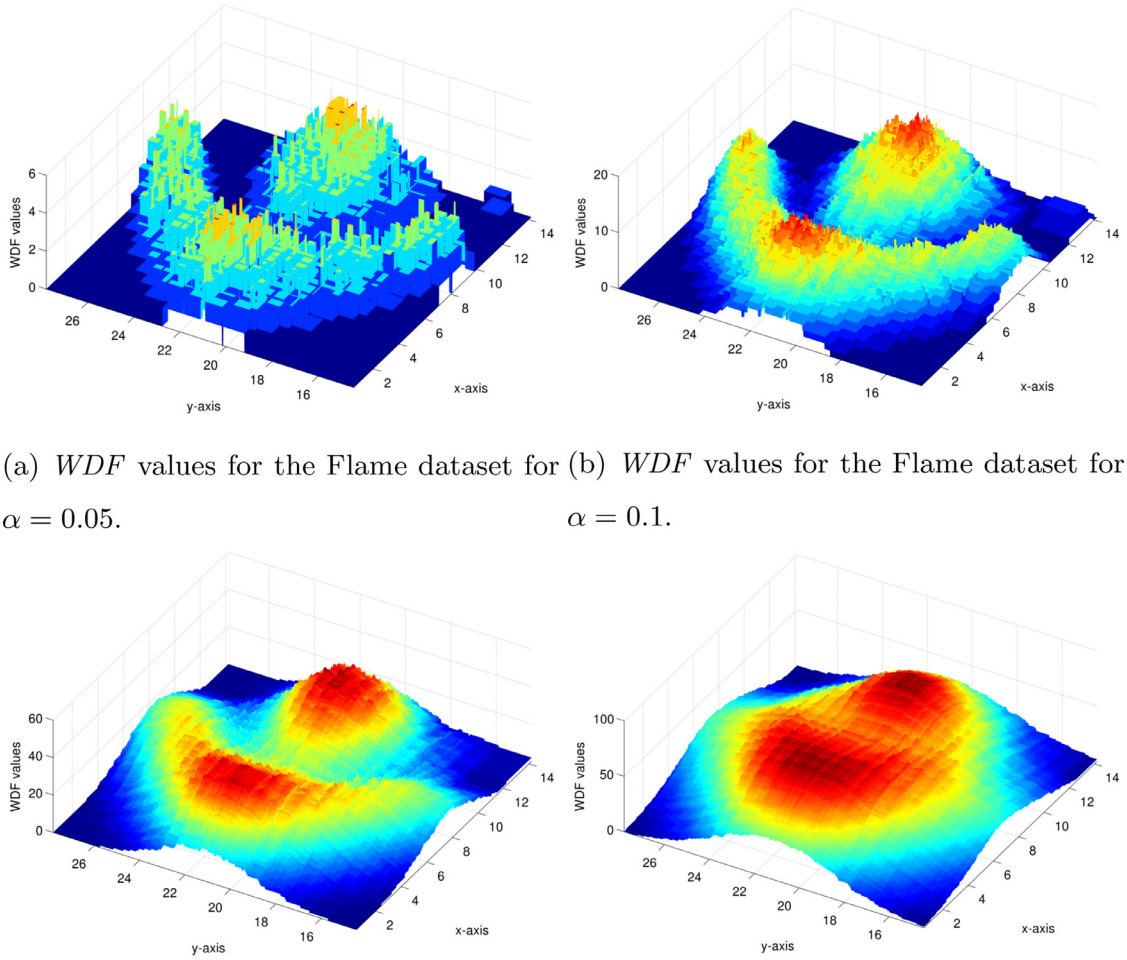
As $WDF_\alpha(z)$ is the number of points of S inside $w_\alpha(z)$, it is obvious that *WDF* is non-negative. Since α remains constant through the comparison of various *WDF* values, *WDF*'s name is justified as the denominator of the actual density remains unaltered and hence the values of the density of a window depend only on the size of $S_{\alpha,\, z}$.

It has been observed that, for low values of α, *WDF* has many local maxima, but whilst the value of α increases, *WDF* reveals the number of local maxima that corresponds to the actual number of clusters [15]. Yet, as α continues to increase, *WDF* becomes smoother and the clusters can not be distinguished. This phenomenon is visualized in Fig. 1. In Fig. 1a small values of α provide a large number of regions, thereby splitting the clusters in smaller clusters, while in Fig. 1d the clusters can not be distinguished.

One approach to find the maxima of *WDF* is the usage of Differential Evolution (DE) [25]. In brief, DE randomly initializes a number (NP) of individuals in the search space, evolves them according to different operators for their mutation and applies crossover to those individuals. The most common operators used for mutation are:

DE1: $v_g^i = x_g^{\text{best}} + F \cdot (x_g^{r_1} - x_g^{r_2})$
DE2: $v_g^i = x_g^{r_1} + F \cdot (x_g^{r_2} - x_g^{r_3})$
DE3: $v_g^i = x_g^i + F \cdot (x_g^{\text{best}} - x_g^{r_1}) + F \cdot (x_g^{r_2} - x_g^{r_3})$
DE4: $v_g^i = x_g^{\text{best}} + F \cdot (x_g^{r_1} - x_g^{r_2}) + F \cdot (x_g^{r_3} - x_g^{r_4})$
DE5: $v_g^i = x_g^{r_1} + F \cdot (x_g^{r_2} - x_g^{r_3}) + F \cdot (x_g^{r_4} - x_g^{r_5})$

where x_g^i is called the parent individual and v_g^i the respective trial vectors, r_1, r_2, r_3, r_4, r_5 are randomly selected integers in $\{1, 2, \ldots, \text{NP}\}$ and $r_j \neq i, \quad j = 1, 2, 3, 4, 5$. x_g^{best} is the best known position observed up to generation g. Finally, $F > 0$ is called the scaling factor. After that, the crossover operation is applied. That is, given a random component j of x_g^i a random number of adjacent components is selected. Those components are selected from v_g^i while the remaining components are selected from x_g^i in order to form an offspring. The offspring will replace its ancestor if it improves its fitness. Fitness will be evaluated as - *WDF*, as DE is a minimization algorithm. This process is repeated for a user defined number of steps (epochs) or until DE reaches a predefined fitness value.



(a) *WDF* values for the Flame dataset for $\alpha = 0.05$.

(b) *WDF* values for the Flame dataset for $\alpha = 0.1$.

(c) *WDF* values for the Flame dataset for $\alpha = 0.2$.

(d) *WDF* values for the Flame dataset for $\alpha = 0.3$.

Fig. 1. Using $\alpha = 0.05$ results in many maxima of *WDF*. Determining the clusters is next to impossible. $\alpha = 0.1$ makes the clusters distinguishable. Small "spikes" still exist. $\alpha = 0.2$ makes the two clusters of the datasets clearly visible as two distinct regions of higher density. $\alpha = 0.3$ on the other hand, provides a single region of higher density, making the two clusters almost indistinguishable.

2.2. A brief description of DeDiCo

DeDiCo is briefly summarized in Algorithm 1. It starts by marking all points as unclustered. Then, the processes of identifying the region of highest density, of moving and enlarging a window covering this region and applying AP to this region (steps 3–13) are repeated until the termination criterion is fulfilled. The termination criterion has two conditions. The main condition is to cluster all points. As this may lead to lengthy runs where only a small number of points is clustered at each iteration, this condition is relaxed to having at most 3 points unclustered. The reasoning for this is to have at least some points so that clustering on those points will be reasonable. Additionally, in order to address the same issue, the second condition is not to exceed a maximum number of iterations.

At each iteration, the number of estimated number of function evaluations used by DE is compared to the number of points remaining unclustered (if-clause at line 3). By utilizing an optimization algorithm, the position of highest density can be located. Unfortunately, this is, computationally, an expensive procedure. Instead, computing the *WDF* value of all unclustered points may be less expensive. Therefore, a choice is made in order to use the

Algorithm 1 A brief overview of **DeDiCo**.

```
1: procedure DeDiCo(S)
2:     while There are still unclustered points or Maximum number of iterations is reached do
3:         if DE function evaluations are more than the number of unclustered points then
4:             Evaluate the WDF value for each point
5:             Select as z the point with the highest WDF value
6:         else
7:             Use DE in order to find the point z with the highest density
8:         end if
9:         Create a window (hyper-rectangle) around z
10:        Move this window
11:        Enlarge this window
12:        Apply AP to the points inside this window
13:        Remove the points in the largest clusters returned by AP
14:    end while
15:    Merge the resulting clusters
16:    Add any still unclustered points to the clusters
17: end procedure
```

smallest number of *WDF* evaluations. As the point of highest density may not be among the points of the dataset, the evaluation of *WDF* for only the points in the dataset, produces an error. Fortunately, the movement and enlargement procedures obsolete this concern. Once the entire dataset is covered by this iterative procedure, the clusters are considered for merging (line 15). Two clusters are merged iff their centers are less that a user-specified distance apart. Finally, any unclustered points are clustered to their nearest cluster.

2.3. An in-depth analysis of DeDiCo

After this brief description, DeDiCowill be analyzed in detail. The first step is to check whether DE will be invoked. It must be mentioned that DeDiCo can utilize any evolutionary optimization algorithm, yet the following description is based on DE. The choice is based on whether it is computationally more expensive to use DE or not. The estimated number of function evaluations is the product of number of individuals times the maximum number of epochs DE is allowed to run. If this product is less or equal than the number of unclustered points in the dataset, then DE is used. In this case, DE searches the space of the dataset in order to minimize the function -*WDF* (line 3). In case the number of points is relative low and DE is not selected for use, the *WDF* of each point is evaluated and the point z with the highest *WDF* value is selected. A window $w_{\alpha,\, s}(z)$ is formed around z. This window is then subjected to movement and enlargement.

2.3.1. The movement procedure

The Movement procedure moves z to coincide with the mean of the points inside the current window. This process is repeated as long as the movements of z add, significantly to the number of points in the window. That is, the number of points added to the cluster by the movement must exceed the initial number of points times a coefficient VT $\in [0, 1]$. VT stands for Variation Tolerance and represents a fraction of points under which, changes in points can be considered negligible.

2.3.2. The enlargement procedure

As a next step, $w_\alpha(z)$ is enlarged in order to capture as much as possible of the detected cluster. This is achieved by enlarging each side of $w_\alpha(z)$ by a fraction MS $\in [0, 1]$. An enlargement over one dimension is accepted only if it adds a significant number of points in the window. That means that the density of the window is not allowed to drop. In order to maintain at least the same density to $w_\alpha(z)$ when enlarging one of its dimension by a fraction of MS, the number of points in it must be also increased by the same fraction. Otherwise, the enlargement is discarded and $w_\alpha(z)$ is enlarged over the next dimension. If at least one enlargement is accepted, the process is repeated on the new window. If no enlargement is accepted over any dimension, the enlargement procedure terminates.

2.3.3. Applying AP

Once window $w_{\alpha,\, s}(z)$ has been moved and enlarged, the points inside it are presented to AP. AP was introduced back in 2007 [29] and has gained a lot of attention since then [18,30–35]. AP is a similarity based iterative algorithm that exchanges "messages" among the points in the dataset until clusters are formed and no longer changed. Additionally, a matrix is formed in order to capture the similarity between each pair of points. As mentioned before, in this contribution the negated Euclidean distance is used. Each diagonal element of this matrix is named self-similarity or preference and represents the suitability of the respective point as a cluster's exemplar.

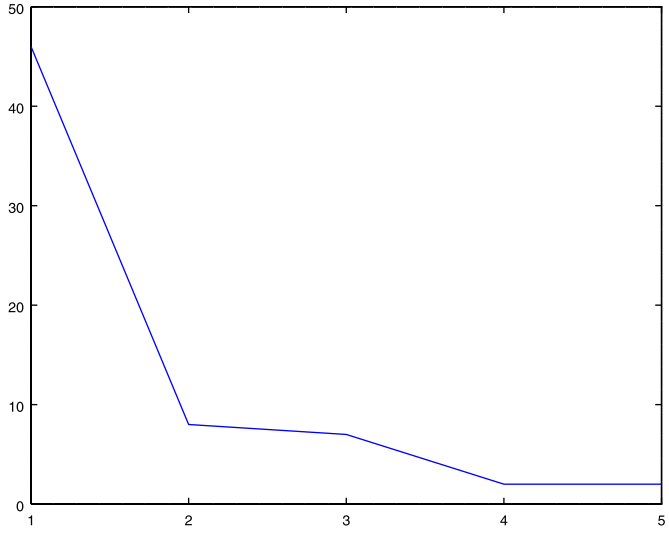


Fig. 2. *WDF* values over DeDiCo iterations for the yeast dataset. x-axis shows the number of iterations of DeDiCo when no iteration limit is posed and y-axis shows the corresponding *WDF* values for the Glass dataset.

AP's preference is controlled by parameter APQ $\in [0, 1]$, which represents the percentile of the sorted similarity matrix. Once AP has produced the clusters for this dataset subset, these are sorted by size and are returned from the largest to lowest up to the point that APRT% points are returned.

As some windows have only a small number of points, using AP may be redundant. For example, clustering windows with only one point, is not reasonable. Therefore, parameter APUT is an integer representing the minimum number of points necessary in order to use AP. In this case all points are clustered together as a single cluster.

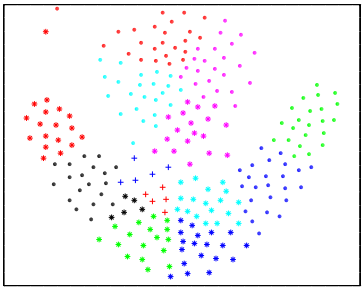
2.3.4. Determining the end of the iterative procedure

If there are not enough points left in the dataset or a maximum number of iteration is reached, this iterative process is terminated. It must be mentioned that this iteration limit is not to avoid infinite loops. After the first experiments, it has been observed that only a small number of iterations provide windows that have a significant number of points. As seen in Fig. 2, the first observed *WDF* value for $\alpha = 0.1$ is 46, yet at iteration 4 it has dropped to 1. Therefore, it is unnecessary to perform the algorithm's iterations and hence they are limited via the Iterations parameter. The approach to cluster the remaining points of the dataset will be described below.

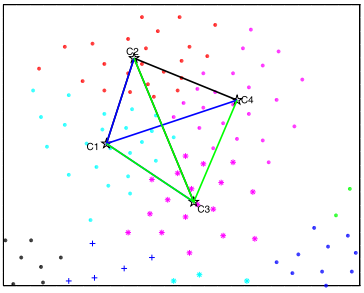
2.3.5. The merging procedure

Once this iterative process is completed, all clusters are considered for merging as described earlier. As AP only considers a part of the complete dataset, the clusters formed by the main iterative loop of DeDiCo are not indicative of the actual clusters. Therefore, these clusters are considered for merging. This is achieved by evaluating the centers (means) of each cluster. As a first step, the distances among the clusters' centers are evaluated. Clusters with centers less than a fraction of the Maximal Diagonal apart are merged. The Maximal Diagonal (MaxDist) is defined as:

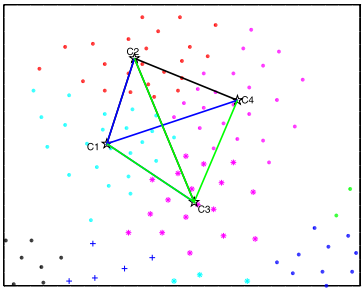
$$\text{MaxDist} = \sqrt{\sum_{i=1}^{d} (H_i - L_i)^2}$$



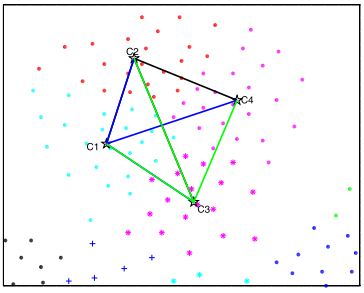
(a) The flame dataset clusters prior to merging. In total 14 clusters are identified.



(b) The upper part of 3a. The black lines represent the distances from C2.



(c) In a similar fashion, the blue lines represent the distances from C1.



(d) In a similar fashion, the green lines represent the distances from C3.

Fig. 3. The Flame dataset after the iterative process of DeDiCo, but prior to merging. The plot focuses on a specific region of the dataset. The cluster means are annotated as stars and named C1, C2, C3 and C4.

where

$$H = \left[\max_{i=1}\left\{S^i\right\}, \max_{i=2}\left\{S^i\right\}, \ldots, \max_{i=d}\left\{S^i\right\}\right]^\top,$$

$$L = \left[\min_{i=1}\left\{S^i\right\}, \min_{i=2}\left\{S^i\right\}, \ldots, \min_{i=d}\left\{S^i\right\}\right]^\top.$$

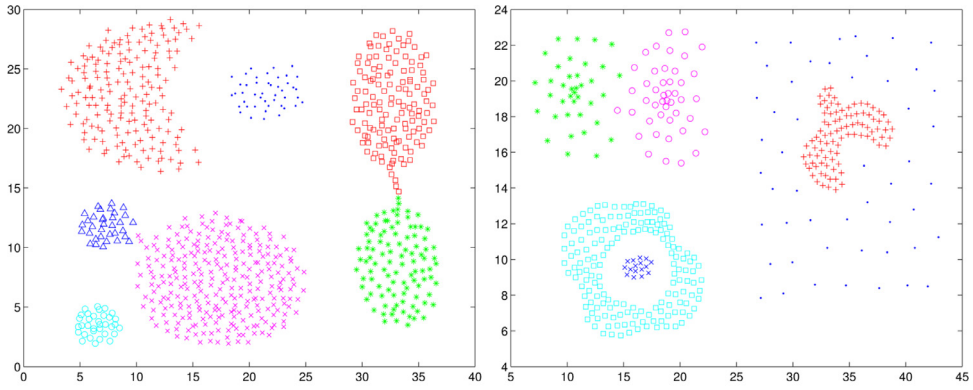
The above mentioned fraction is the parameter MergeCoefficient (MC $\in [0, 1]$). If a cluster is not close enough to any other cluster, then it is given a new id. If it is close enough to other clusters, the choice of its id is not straightforward. In case that none of the clusters to be merged has an id, a new id for all of them is generated and assigned to them. Finally, if at least one of the other clusters has already an id, the smallest id is selected and assigned to all of the clusters to be merged as well as any other cluster that may share an id with those clusters. For the example of Fig. 3, the distances for cluster centers C1, C2, C3 and C4, annotated in 3 b, are 2.82, 2.93 and 4.84, while MaxDist is 19.13. The ratios of the distances over MaxDist are 0.15, 0.15 and 0.25. This means that setting MC to values lower than 0.15, the cluster represented by C2 will not be merged with any other clusters. On the other hand, setting MC to values higher than 0.25 will result in merging all the clusters, as they will all receive the same id as the upper cluster. Setting MC as 0.19 will merge clusters represented by C1 and C4 with C2. When the distance between C1 and C3 is examined (Fig. 3c), C1 will be merged and hence all four clusters will be assigned to the same id. Finally, the points that have not been clustered up to this point, are appended to a cluster depending on which cluster's center is closer to them.

3. Evaluation process

3.1. Datasets

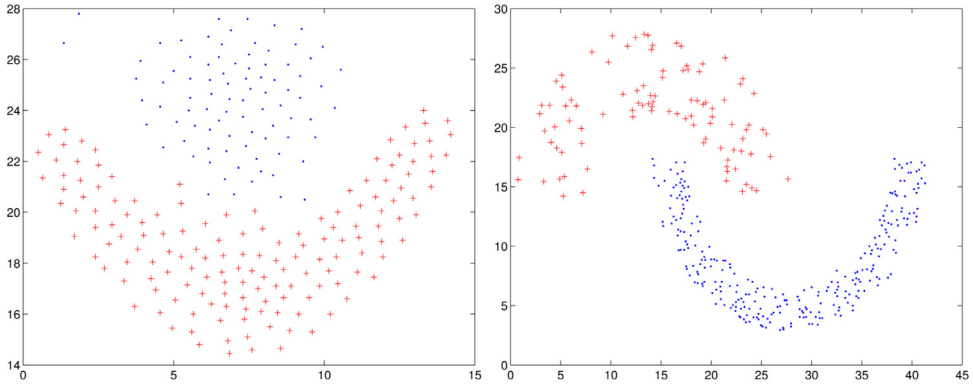
The used clustering algorithms are compared on synthetic as well as real world datasets. The synthetic datasets were retrieved from https://cs.joensuu.fi/sipu/datasets/. All of them have been used in other publications.

These datasets were selected due to the fact that they provide challenges to clustering algorithms. For example, the Aggregation dataset (Fig. 4a, [36]) has two clusters which have points that form a line between them, making them almost indistinguishable. The Compound dataset (Fig. 4b, [37]) has two challenges. On the first hand, two clusters on the top left are close together. Although this may pose a problem in the proper configuration of the algorithms, it should not prevent an algorithm from identifying these two clusters. On the other hand, on the right of the image as well as on the lower left part, two clusters are found, one inside the other. Additionally, on the right two clusters, only the density differs. The Pathbased dataset (Fig. 6, [38]) and the Flame dataset (Fig. 5a, [39]) have similar issues. In these datasets there is not a single line of points that confuses clustering algorithms but rather the close proximity of some clusters, making a good parameter choice vital for the success of an algorithm. Finally, the R15 dataset (Fig. 6b, [40]) has some of its clusters well separated. Unfortunately, some of them are close and two of them have mixed points. The points marked as members of one cluster reside among points belonging to other clusters and are hardly possible to be clustered correctly.



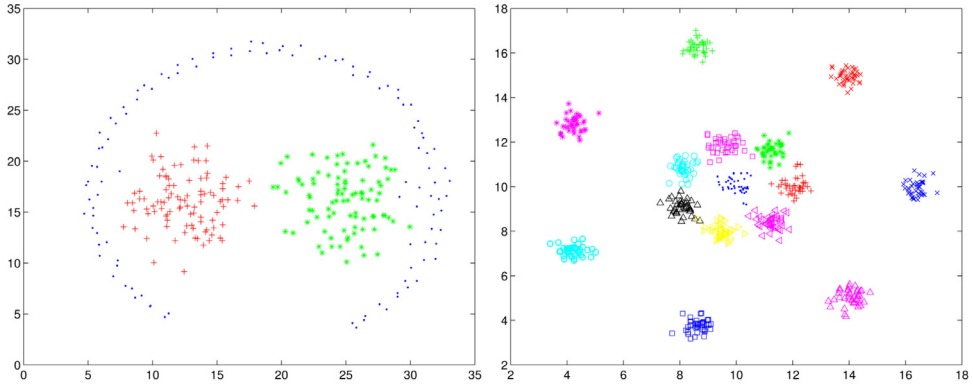
(a) The Aggregation dataset (b) The Compound dataset

Fig. 4. The datasets retrieved from Joensuu (1/4).



(a) The Flame dataset (b) The Jain dataset

Fig. 5. The datasets retrieved from Joensuu (2/4).



(a) The Pathbased dataset (b) The R15 dataset

Fig. 6. The datasets retrieved from Joensuu (3/4).

The considered real datasets are among the most commonly used datasets and were retrieved from the UCI repository (http://archive.ics.uci.edu/ml/). The major challenge that arises is their higher dimensionality. The Iris dataset [41] is a 4-dimensional dataset with sepal length, sepal width, petal length and petal width as arguments. Each of the 3 classes responds to one of three types of the plant Iris. The Wine dataset is a 13-dimensional dataset which features represent measurements from a chemical analysis of wines belonging to three cultivars. The Glass dataset is a 9-dimensional dataset containing glass attributes that should be sufficient to identify the type of glass. The Wisconsin dataset [42] is a 9-dimensional dataset with cell characteristics for identifying breast cancer cells. Banknote is a dataset based on images of genuine and forged of banknote-like specimens [43]. Its 4 features are based on wavelet transformed images as well as the images' entropy. The final utilized dataset is called Yeast and uses 8 attributes in order to classify yeast proteins into 10 classes [43,44].

3.2. Computational experiments

There are two methodologies to evaluate the results of a clustering algorithm. If the information of each point's cluster assign-

ment (ground truth) is at hand, one may use external validation measures, usually derived from classification. Yet, such knowledge is rarely available with the exception of some synthetic datasets or actual preprocessed datasets. On the other hand, internal validation measures [45–47] require no knowledge of the actual clustering. Unfortunately, such measures can rarely be used for evaluating the performance of clustering algorithms in the case of randomly shaped sets as they favor a specific form of clusters [48]. Therefore, the external criterion V-measure [49] was used in order to evaluate the performance on a set of datasets where each point's cluster assignment is known. The calculation of V-measure is based on the notions of Homogeneity and Completeness as introduced by Rosenberg and Hirschberg [49]. Homogeneity measures the degree by which the algorithm's clusters consist solely of members of a single cluster of the ground truth. On the other hand, Completeness measures whether the members of a cluster of the ground truth are clustered together into a single cluster or not.

DeDiCo is compared to the algorithms *k*-means, DBSCAN and AP as well as the algorithm introduced by Rodriguez and Laio [17] which will be abbreviated as FSDP. The motivation for using those algorithms is the attention these algorithms have received. Despite its preference to spherical clusters, *k*-means could be the most used algorithm in data clustering. Therefore, its results are presented as benchmark values. DBSCAN has not been studied as much as *k*-means. Yet, it is a well-known density based algorithm, capable of detecting arbitrary shaped clusters as well as outliers. Furthermore, AP is used as it is a component of DeDiCo and therefore, it is used to test the improvement contributed from DeDiCo's other procedures. Finally, FSDP can be considered as a state of the art algorithm.

k-means, DBSCAN, AP and DeDiCo were run 100 times. *k*-means uses random initialization for the cluster centers. DeDiCo can be a stochastic algorithm. Therefore, it is necessary for it to run multiple times. AP, on the other hand, is deterministic by its nature. Yet, its implementations add some noise in order to avoid degenerate cases. As a consequence, it was deemed necessary to run AP 100 times as well. Finally, DBSCAN may produce varying results if presented with a different succession of points. Although this phenomenon is considered rare, DBSCAN also ran 100 times, each time with a different permutation of the points of the dataset so that the points in the dataset are presented to it in a different order. As V-measure does not handle outliers, the points not clustered (outliers) were monitored. All results reported have no outliers.

k-means does not require any parameter aside from the number of clusters. Therefore, k is provided with the known value of the cluster number. Similarly, FSDP was provided with parameters that result in the predefined number of clusters. To be precise, the authors of [17] provide a method of determining the algorithm's two parameters, namely ρ and δ. ρ expresses the density around a point and δ the distance to the nearest point of increased density. As the authors advise, a $\rho - \delta$ plot is created and the ρ_0 and δ_0 values are selected so that for only k points it holds that $\rho \geq \rho_0$ and $\delta \geq \delta_0$, where k is the number of clusters in the dataset. DBSCAN ran on a large range of values for the radius of the hypersphere. More precisely, for radii ranging from 0.0001 to 0.5 of the largest distance among points, 100 permutations of the dataset were created. This was done in order to address the fact that changing the order the points are presented to DBSCAN may alter the clusters returned. The desired number of points in the hypersphere is kept constant as advised Ester et al. [1]. The reason for testing a large number of radii is to ensure that DBSCAN is tested exhaustively. Its relatively low running times make this feasible. AP ran for a variety of preference values based on different percentiles of the values of the similarity's matrix. Fig. 8 depicts the similarities measured for the flame dataset. For example, setting the preference to $q = 0.1$, corresponds to the 10th percentile's value. In the case of the Flame dataset, this value is −94.40. For the experiments, the 0th, 10th, ..., 100th percentile were used. Table 2 shows the values for each percentile for the Flame dataset.

Table 1
An exhaustive list of all parameters of DeDiCo. NP is the number of individuals used by DE. itermax is the number of epochs for DE. *F* is the mutation parameter for DE. CR is DE's crossover parameter, named probability of recombination. All other parameters are described in the text.

Parameter	Abbreviation
NP	–
itermax	–
strategy	–
F	–
CR	–
Iterations	–
Modification Step	MS
Merge Coefficient	MC
Variation Threshold	VC
Alpha	α
AP Use Threshold	APUT
AP Quantile	APQ
AP Return Threshold	APRT

Table 2
Percentile values for the Flame dataset.

Percentile	0th	10th	20th	30th	40th	
Value	−217.81	−94.40	−72.43	−57.36	−45.31	
Percentile	50th	60th	70th	80th	90th	100th
Value	−35.12	−25.92	−17.47	−10.26	−4.28	0

DeDiCo has 13 parameters and so fine-tuning can be a laborious task. Therefore, through the trial and error approach a small subset of values for each parameter was selected. Over these values, exhaustive search was performed. Initial experimentation determined the range of those values. By selecting as few as 3 values for each parameter would result in 3^{13} experimental setups for each dataset. Additionally, DeDiCo is a stochastic algorithm and hence must be executed 100 times for each of the 13 datasets. This would result in an unfeasible number of experiments. Therefore, experimental experience must be used to create smaller ranges for each parameter and even set some of them to specific values. Through this process, successfully determining the appropriate number of clusters was also a contributing factor, aside of achieving high V-measure values.

The first observation was that the majority of DE's parameters do not have an important impact on DeDiCo's performance. Therefore, DE's scaling factor was set to 0.6 and the crossover constant was set to 0.8. The number of parameters was 10-fold the dimensionality of each dataset. Parameters such as strategy could be set to any value. The main reason for this is that although these parameters affect the performance of DE, DeDiCo utilizes DE only as a part (if at all) and its other mechanisms can overcome a possibly compromised performance of DE. The main parameters that do have an effect on DeDiCo prove to be MS, MC, VT, α and the AP related parameters APQ and APRT.

By examining the parameters for which DeDiCo achieved its best performance on each dataset, the following remarks can be made. MS ranged in [0.01, 0.06] with the exception of Glass and Yeast. Therefore, an initial value in this range is suggested. As MS expresses the change between consecutive enlargements, its low values are well justified by the fact that larger values would cause a window enlargement to incorporate a nearby cluster and hence relying more on AP to distinguish those clusters.

MC also shows a stability as it ranged in [0.05, 0.2] with 0.1 providing the best results for 5 and 0.15 for another 3 datasets. For MC the range [0.05, 0.2] is suggested as an initial range. MC

represents the eagerness of the algorithm to merge nearby clusters. Setting MC to much larger values would merge clusters that are further apart, recanting the contribution of the previous steps, especially AP. It is suggested to show moderation when increasing MC.

VT ranged in [0.0001, 0.0008] with the exception of Flame, where VT was 0.1. In 7 datasets VT was 0.0001. As this parameter encaptures the tolerance in number of points changed through a movement or enlargement step, its value depends on the number of points in the dataset as well as on how precise those two functions may be. Should those two steps terminate only when almost no change in the number of points is observed, VT should be set to a low value. If the number of points rises significantly, this value may need to be reduced in order to have the same effect. On the other hand, by setting VT to larger values, the user allows for a brief movement/enlargement procedure and relies on AP and merging to refine the clustering results. This can be translated to compromising performance in order to improve regarding the computational effort. Empirically, VT is best refined by changing its value by powers of 10 and further refine its values once the appropriate power has been determined.

One of the most influential parameters, α, usually was 0.1 or close to 0.1 for the synthetic datasets, with only exception the Pathbased dataset, where it was 0.6. Such a large value indicated a difficulty of DeDiCo to rely on movement and enlargement alone in order to capture the clusters in this dataset. On the other hand, on real datasets, α had values over 0.4. A possible explanation is that with the rise of dimensionality and not respective rise to the number of points, the increased sparcity starts to influence the contribution of the *WDF* function, as it measures density. Therefore, larger values are needed so that larger areas are probed in order to provide a range of *WDF* values. One exception is the Banknote dataset, where the number of points is among the largest used and the dimensionality of 4 is relatively small. For this dataset α was 0.15. Another exception is the Glass dataset. Yet, for this dataset MS is the largest used. Therefore, α's small initial value is compensated by MS's large value. That is, DeDiCo starts with a small α value and significantly increases it to reach appropriate sizes.

Finally, the values of ARQ and APRT are the hardest to determine. Their values ranged in their entire validity range. Yet, ARQ provided the best results for values larger than 0.6 for the majority of the datasets. Notably, for no dataset its maxima was used. Unfortunately, a similar conclusion cannot be drawn for APRT, whose values are distributed fairly uniform in [0.1, 0.9] and is unrelated to APQ. Other parameters are mainly used to prevent the algorithm from degenerate states, like the APUT, which was fixed to 6 for all experiments.

Table 3
V-measure and time used for the synthetic datasets.

	V-measure			CPU time (sec)		
	Max	Mean	Std	Max	Mean	Std
Aggregation						
AP	0.8671	0.8671	0	8	4.1404	2.4810
k-Means	0.8765	0.6957	0.0705	6	4.3308	1.4828
DBSCAN	08894	0.8894	0	**4**	**1.2549**	1.3212
DeDiCo	0.9924	0.9924	0	10.5956	10.3931	**0.0740**
FSDP	–	**0.9957**	–	–	30.7613	–
Compound						
AP	0.7713	0.7713	0	4	1.9031	1.2923
k-Means	0.8197	0.5995	0.1048	5	3.1900	1.9871
DBSCAN	0.8061	0.8061	0	2	**0.7701**	0.7704
DeDiCo	0.8523	**0.8523**	0	**1.7080**	1.5953	**0.0238**
FSDP	–	0.58770	–	–	7.7874	–
Flame						
AP	0.5384	0.5384	0	3	1.3406	1.1253
k-Means	0.4685	0.4262	0.0282	**1**	0.4938	0.5010
DBSCAN	0.8896	0.7991	0.0910	2	1.6724	0.4785
DeDiCo	0.9359	**0.9359**	0	1.3119	1.2323	**0.0130**
FSDP	–	0.6328	–	–	**0.0528**	–
Jain						
AP	0.4396	0.4396	0	7	4.0675	2.1805
k-Means	0.3690	0.3611	0.0067	**1**	**0.5294**	0.4998
DBSCAN	0.9111	0.7713	0.0456	5	1.9922	1.7334
DeDiCo	1	**1**	0	2.6640	2.5069	**0.0246**
FSDP	–	0.50522	–	-	6.74551	–

Table 4
V-measure and time used for the synthetic datasets.

	V-measure			bf CPU time (sec)		
	Max	Mean	Std	Max	Mean	Std
Pathbased						
AP	0.5525	0.5525	0	9	5.6483	3.0206
k-Means	0.5489	0.5376	0.0165	**2**	1.0067	0.9451
DBSCAN	0.7615	0.6214	0	13	4.2694	3.5770
DeDiCo	0.6652	0.6586	0.0058	2.8874	2.4381	**0.1048**
FSDP	–	**0.8794**	–	–	**0.0929**	–
R15						
AP	0.9942	**0.9942**	0	14	6.9859	4.3308
k-Means	0.9395	0.7874	0.0497	14	8.0765	4.6688
DBSCAN	0.9128	0.9050	0.0142	12	5.8372	3.5347
DeDiCo	0.7372	0.7372	0	**6.4003**	**5.7274**	**0.3785**
FSDP	–	0.9942	–	–	18.0879	–
Spiral						
AP	0.4936	0.4936	0	26	13.4512	7.9564
k-Means	0.0013	0.0006	0.0002	**2**	1.3227	0.9480
DBSCAN	1.000	**1.000**	0	**2**	**1.0097**	0.8143
DeDiCo	0.4704	0.4704	0	2.1254	2.0432	**0.0155**
FSDP	–	**1.000**	–	–	4.9603	–

4. Results and discussion

The V-measures for all algorithms are reported in Tables 3–6. The results prove that DeDiCo outperforms the compared algorithms in most datasets. More specifically, on most synthetic datasets, DeDiCo is not overwhelmed by the difficulties in each such dataset, while even reaching a perfect score for the Jain dataset. On the contrary, AP is significantly outperformed by DBSCAN as well. R15 is an exception. Here AP provides the best clustering by reaching an almost perfect score. Finally, Spiral (Fig. 7a) is a unique dataset considering the layout of its clusters. Besides DBSCAN and FSDP, which score a perfect score, all other algorithms struggle. Especially, *k*-means fails, as expected, to provide a satisfactory clustering.

Moving to real-life datasets, DeDiCo still manages to surpass the other algorithms in 3 datasets. *k*-means, AP and DBSCAN manage to provide the best clustering in only 1 dataset each. In these datasets, DeDiCo is still able to provide the second best clustering.

Finally, DeDiCo may not be the fastest algorithm, yet its standard deviation is the lowest in 11 out of 13 datasets. Concerning CPU time, *k*-means, closely followed by DBSCAN and FSDP, are by far the best performing algorithms. Unfortunately, in the case of *k*-means this does not correspond to an elevated performance, as its V-measure mean is significantly inferior, with the exception of the Wisconsin dataset. DBSCAN's times are also very low, yet, they have a larger standard deviation. FSDP can achieve impressive CPU times which, occasionally, are accompanied by good results.

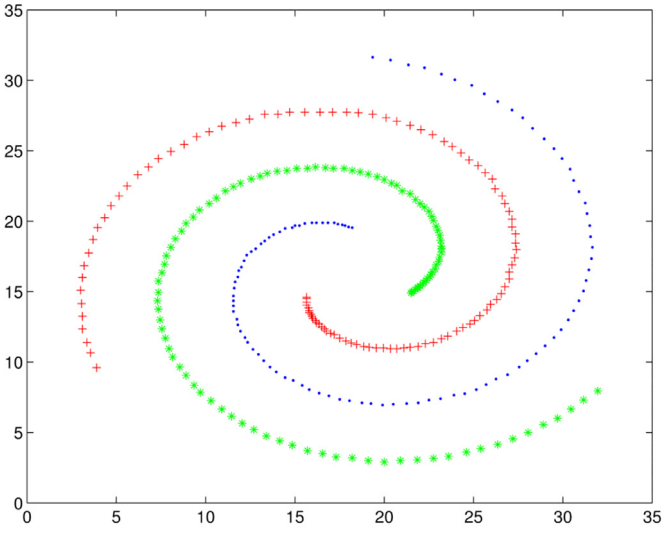
The present experiments above indicate that DeDiCo can outperform other clustering algorithms on a variety of datasets. From synthetic datasets like Aggregation and Jain, to the well known real life Iris dataset, DeDiCo improves the performance. Additionally, its large number of parameters make it versatile and allow DeDiCo to adapt to a large number of different kind of problems. Its running

Table 5
V-measure and time used for the real-life datasets.

	V-measure			CPU time (sec)		
	Max	Mean	Std	Max	Mean	Std
Wine						
AP	0.4241	0.4241	0	**2**	1.1290	0.8068
k-Means	0.4484	0.4131	0.0796	**2**	1.1173	0.8021
DBSCAN	0.3032	0.1707	0.0538	10	2.0731	3.3768
DeDiCo	0.4786	**0.4786**	0	3.0255	2.1801	**0.1020**
FSDP	–	0.4863	–	–	**0.0278**	–
Iris						
AP	0.7980	0.7980	0	2	0.8943	0.7494
k-Means	0.7582	0.6994	0.1128	2	1.3245	0.9490
DBSCAN	0.7337	0.7337	0	**1**	0.6624	0.4743
DeDiCo	0.8334	**0.8174**	0.0198	2.4691	1.9985	**0.1938**
FSDP	–	0.7650	–	–	**0.0178**	–
Glass						
AP	0.4689	0.4689	0	10	2.3219	2.8201
k-Means	0.4277	0.2903	0.0822	5	1.6279	1.4212
DBSCAN	0.3976	0.3125	0.6950	4	0.9490	0.8438
DeDiCo	0.4968	**0.4710**	0.0137	**2.4282**	2.0469	**0.1378**
FSDP	–	0.4326	–	–	**0.0525**	–
Wisconsin						
AP	0.4485	0.4459	0.0027	8	2.3865	2.2030
k-Means	0.7546	**0.7512**	0.0034	**1**	**0.3377**	**0.4733**
DBSCAN	0.7122	0.4057	0.2103	4	1.4549	0.8583
DeDiCo	0.6646	0.6078	0.0146	10.8226	7.7820	1.5362
FSDP	–	0.0548	–	–	21.6805	–

Table 6
V-measure and time used for the real-life datasets.

	V-measure			CPU time (sec)		
	Max	Mean	Std	Max	Mean	Std
Banknote						
AP	0.3589	0.3589	0	18	8.0021	5.5438
k-Means	0.0303	0.0303	0	**1**	**0.3365**	**0.4727**
DBSCAN	0.7289	**0.7108**	0.0109	10	4.2713	2.1237
DeDiCo	0.6756	0.6660	0.0335	43.8663	38.6501	3.5987
FSDP	–	0.3544	–	–	5.6776	–
Yeast						
AP	0.3835	**0.3833**	0	1461	732.0459	421.3157
k-Means	0.2775	0.1882	0.0424	**9**	6.0647	2.8813
DBSCAN	0.0544	0.0430	0.0049	19	**1.9149**	3.4244
DeDiCo	0.2912	0.2889	0.0012	65.6905	62.7165	**1.1823**
FSDP	–	0.0657	–	–	101.839	–



(a) The Spiral dataset

Fig. 7. The datasets retrieved from Joensuu (4/4).

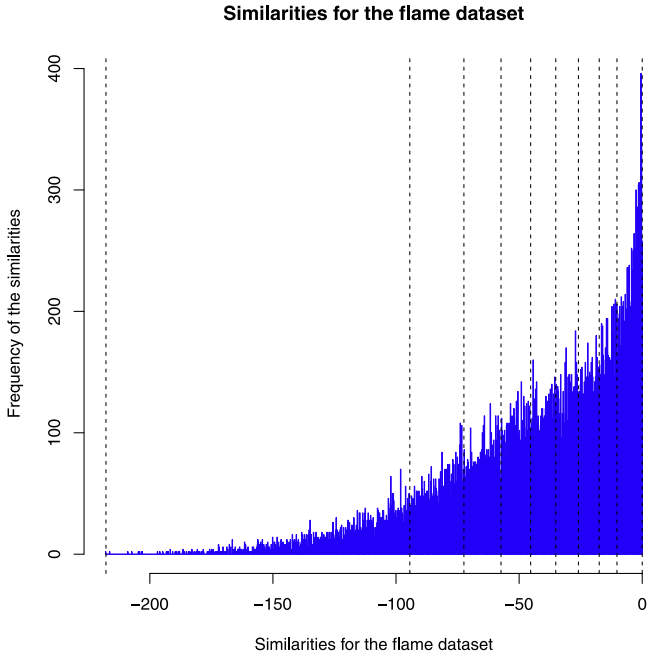
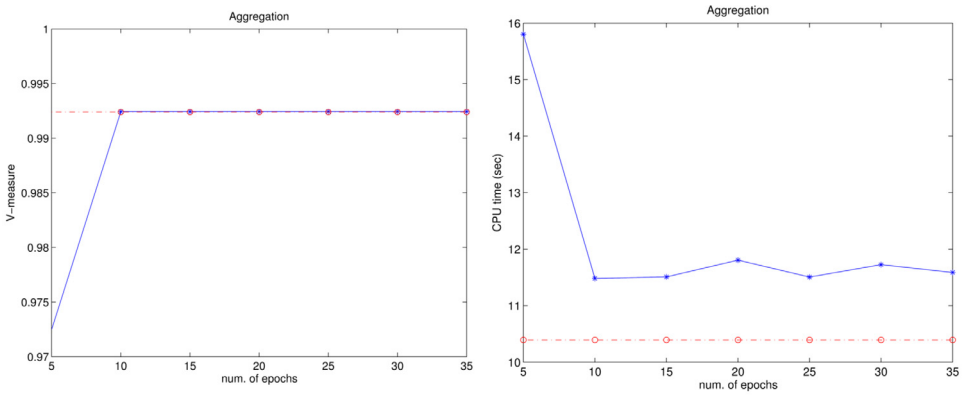


Fig. 8. Histogram for Flame datasets similarities. The horizontal axis represents the similarity values, while the vertical axis has their frequency. Vertical lines represent the 0th, 10th, . . . , 100th percentile.

times can also be adapted by allowing the invoking of DE or not. On large datasets the use of DE or any other evolutionary optimization algorithm should be encouraged so that the number of function evaluations is kept to a minimum. On the other hand, on small datasets, DE should be bated in order to keep running times as low as possible.

Three datasets have been selected in order to communicate the influence of DE into DeDiCo. According to a common practice, DE's parameter NP was set to be 10 times the dataset's dimensionality. Simultaneously, the number of epochs varied, as seen in Figs. 9–11. It is evident than the usage of DE cannot surpass the performance of DeDiCo without it. In general, its performance is lower with DE and only for the Aggregation dataset it can match its performance without DE. On the other hand, times can be influenced by the usage of DE. For the Aggregation dataset (Fig. 9b), DeDiCo with DE is significantly slower. On the contrary, for the Pathbased dataset (Fig. 10b), DeDiCo needs approximately the half time with DE in comparison to DeDiCo without DE. Finally, in the case of the Banknote dataset (Fig. 11b), the number of epochs influences DeDiCo's times. An explanation for the decrease of DeDiCo's time when the number of epochs increases could be that the number of iterations decreases, as dataset's regions with higher density are visited and therefore, the total number of iterations decreases.

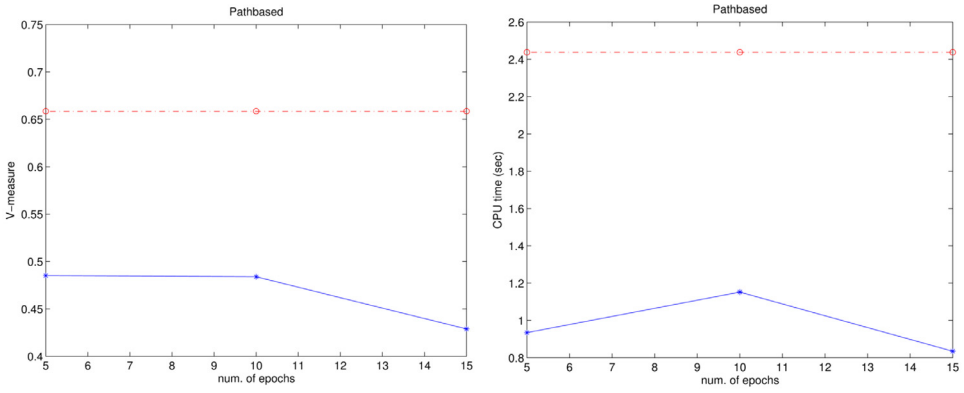
Regarding the other steps of DeDiCo, a comparison was made on some of the datasets. Aggregation, Pathbased and Flame were selected due to the fact that they have clusters in relatively close proximity which can be difficult to distiguish. Jain was used as it is an easy dataset, which should be easily addressed by a clustering algorithm. Iris is a dataset commonly used for testing clustering performance. Wine, Glass and Wisconsin datasets were selected due to the fact that they contain only few points in relation to their dimensionality. The four main steps of DeDiCo were removed in order to encapture their influence. Movement, enlargement, merging and AP were removed and the resulting modifications run on the parameters which resulted in the best results, as presented in Tables 3–6. Obviously those values were selected in order to provide the best results for the original proposed algo-



(a) V-measure values on the Aggregation dataset when using DE.

(b) CPU times in sec. for the Aggregation dataset when using DE.

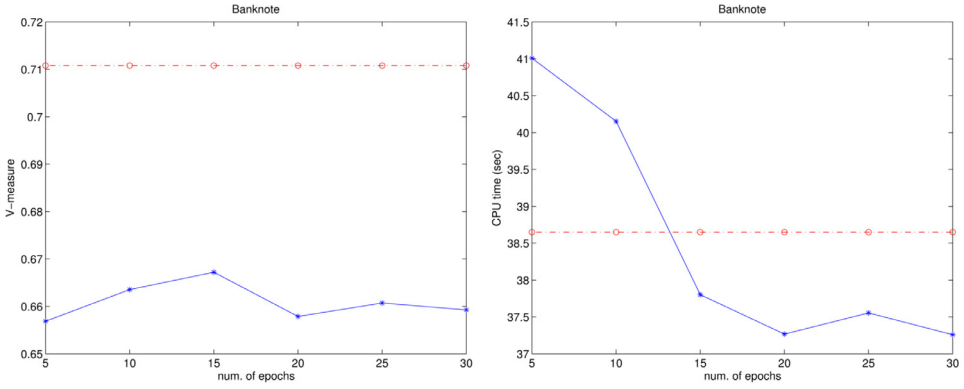
Fig. 9. Aggregation dataset: Red line indicates the mean performance when DE is not utilized, blue line is when DE is utilized. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) V-measure values on the Pathbased dataset when using DE.

(b) CPU times in sec. for the Pathbased dataset when using DE.

Fig. 10. Pathbased dataset: Red line indicates the mean performance when DE is not utilized, blue line is when DE is utilized. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) V-measure values on the Banknote dataset when using DE.

(b) CPU times in sec. for the Banknote dataset when using DE.

Fig. 11. Banknote dataset: Red line indicates the mean performance when DE is not utilized, blue line is when DE is utilized. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 7
Influence of DeDiCo's various steps in its performance. DeDiCo is the proposed algorithm's performance. DeDiCo without movement indicates that the movement step was deactivated from the proposed algorithm. Similarly, the enlargement, the merging and the AP step were deactivated. Values under 10^{-4} are presented as zero.

V-measure for DeDiCo and its modifications						
		DeDiCo As proposed	DeDiCo Without movement	DeDiCo Without enlargement	DeDiCo Without merge	DeDiCo Without AP
Aggregation	Max	0.9924	0.9198	0	0.6664	0
	Mean	0.9924	0.9198	0	0.6664	0
	Std	0	0	0	0	0
Pathbased	Max	0.6630	0.6630	0.5714	0.4681	0.0598
	Mean	0.6630	0.6630	0.5714	0.4681	0.0598
	Std	0.0002	0.0002	0	0	0
Flame	Max	0.9359	0.8523	0.2360	0.3373	0
	Mean	0.9359	0.8523	0.2360	0.3373	0
	Std	0	0	0	0	0
Jain	Max	1	1	0	0.3644	0
	Mean	1	1	0	0.3644	0
	Std	0	0	0	0	0
Iris	Max	0.8334	0.8334	0.7837	0.5692	0.6699
	Mean	0.8174	0.8205	0.7800	0.5692	0.6699
	Std	0.0137	0.0117	0.0032	0	0
Wine	Max	0.4786	0.4786	0.4292	0.3484	0.0623
	Mean	0.4786	0.4786	0.4292	0.3484	0.0623
	Std	0		0	0	0
Glass	Max	0.4968	0.4949	0.1097	0.4047	0
	Mean	0.4710	0.4688	0.0965	0.4047	0
	Std	0.0137	0.0117	0.0032	0	0
Wisconsin	Max	0.6646	0.5815	0.5613	0.2598	0.5532
	Mean	0.6078	0.5815	0.5315	0.2523	0.0163
	Std	0.0146	0	0.0163	0.0031	0

rithm and not necessarily for its modifications. Yet, the purpose of this comparison is to indicate the influence of each step in the final result and not provide the best performance for each modification. The results are presented in Table 7. From this Table the limited influence of the movement procedure must be noted. In the case of the Jain, the Pathbased and the Wine datasets, movement did not provide any improvement to the final result. For the remaining datasets, the movement step did contribute to the final performance. On the other hand, the merge step proved its significance. Even more influential are the steps of enlargement and AP as when they are removed, the resulting algorithm cannot provide a clustering with a V-measure higher than 10^{-4} for many datasets. Once more it must be stressed that the modifications ran on the parameters that provided the best clustering in the comparative experiments and they were not reconfigured to provide the best clustering for the modifications. This depicts the contribution of those steps on the best result.

On the used datasets here, DeDiCo clearly outperforms its competition. As a second best, DBSCAN, in general, is a good choice as it provides a constantly high performance. AP outperforms the other algorithms in only three datasets.

A possible explanation for the predominance of DBSCAN can be the nature of the Flame and the Pathbased datasets. They both contain clusters in very close proximity. On the other hand, the Spiral dataset provides a significant challenge since all three clusters are similar in density and shape. DBSCAN was able to cluster all points perfectly. This highlights its increased ability to capture arbitrary shaped clusters. Selecting DeDiCo over AP is a rational choice, as it requires significantly less memory by applying AP only to a fraction of the dataset. One may argue that in some cases AP may run on almost the entire dataset. Yet, such cases may be avoided by better parameter selection. Especially α, *VT* and *MS* seem to significantly affect the final window size to which AP is applied. Moderate values for those parameters result in relatively small window and hence allow fast executions of AP with low memory requirements.

5. Conclusion – Future work

The algorithm presented in this work has the ability to modify its execution based on the dataset size. It can decide at any given step to utilize an evolutionary optimization algorithm such as DE or not. Moreover, DeDiCo uses both the notion of density as well as the notion of distance. Yet, the algorithm does not combine them into a single metric, but uses them separately. Thereby, the benefits of each approach can be exploited, without compromising the clustering process by creating a new concept that merges those notions.

DeDiCo has proven to outperform the algorithms that were used in this work, in the majority of the experiments. Despite being a stochastic algorithm, its performance has a remarkably low variance. This also holds for its running times. As DeDiCo outperforms AP in the majority of the datasets, this algorithm does not simply incorporate a successful algorithm, but provides a novel framework that exploits AP and more specifically its distance-based approach to create more refined clusters. All datasets used have been selected as each of them introduces difficulties regarding clustering. The majority of the synthetic datasets has issues regarding the proximity of the clusters. Yet, DeDiCo enables it to adjust to each dataset and therefore outperforms the other algorithms. In addition, the real-life datasets on which DeDiCo provides better V-measure values, have a small number of points regarding their dimensionality. On datasets with a high number of points, the algorithm's ability to adapt its behavior allows it to have a smaller computational cost in comparison to other algorithms that reside on interactions among all points, like AP. On smaller datasets, DeDiCo is computationally more efficient than evolutionary clustering algorithms. Density and distance based

clustering algorithms are affected by the curse of dimensionality if there are not sufficient points [50]. DeDiCo was successfully tested on datasets where the dimensionality is relatively high regarding the number of points such as the wine, the Wisconsin, the yeast and the glass datasets.

However, still some issues remain to be addressed. Mainly, the fact that it has a notable number of parameters which posed a significant problem in parameter fine-tuning. Those contribute to its success, as it can adapt to perform well on each given dataset. Therefore, special care must be taken in order to provide a framework for parameter tuning. Some parameters may be ignored in this process. For example, in a small dataset, parameters regarding DE may be left with their default values, as DeDiCo can select not to use DE as it would be computationally more expensive.

DeDiCo was compared on both synthetic and well analyzed real-life datasets. Since clustering is a topic with significant research activity, it is must be expected that DeDiCo will be compared with novel methods. Further studies will be conducted by our groups in order to investigate the algorithm's performance in open real-life problems as arise in fields such as clustering in medical images and molecular expression data as well.

Acknowledgments

We thank the anonymous reviewers for their constructive comments, which helped us to improve the manuscript.

References

[1] M. Ester, H.P. Kriegel, J. Sander, 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, 1996, pp. 226–231.
[2] Y. Zhu, K. Ming Ting, M.J. Carman, Density-ratio based clustering for discovering clusters with varying densities, Pattern Recognit. 60 (2016) 983–997.
[3] K. Mahesh Kumar, A. Rama Mohan Reddy, A fast DBSCAN clustering algorithm by accelerating neighbor searching using groups method, Pattern Recognit. 58 (2016) 39–48.
[4] D.K. Tasoulis, M.N. Vrahatis, The new density function for efficient evolutionary unsupervised clustering, in: IEEE Congress on Evolutionary Computation, CEC 2005, 3, IEEE Press, 2005, pp. 2388–2394.
[5] Y. Kim, K. Shim, M.-S. Kim, J.S. Lee, DBCURE-MR: An efficient density-based clustering algorithm for large data using mapreduce, Inf. Syst. 42 (2014) 15–35.
[6] A. Bryant, K. Cios, RNN-DBSCAN: a density-based clustering algorithm using reverse nearest neighbor density estimates, IEEE Trans. Knowl. Data Eng. 30 (6) (2018) 1109–1121.
[7] Z. Liang, P. Chen, Delta-density based clustering with a divide-and-conquer strategy: 3DC clustering, Pattern Recognit. Lett. 73 (2016) 52–59.
[8] G. Karypis, H.E. Han, V. Kumar, CHAMELEON: a hierarchical clustering algorithm using dynamic modeling, Computer 32 (1999) 68–75.
[9] A. Kobren, N. Monath, A. Krishnamurthy, A. McCallum, A hierarchical algorithm for extreme clustering, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD '17, 2017, pp. 255–264.
[10] R.L. Cilibrasi, P.M. Vitányi, A fast quartet tree heuristic for hierarchical clustering, Pattern Recognit. 44 (3) (2011) 662–677.
[11] S. Li, W. Li, J. Qiu, A novel divisive hierarchical clustering algorithm for geospatial analysis, Int. J. Geo-Inf. 6 (1) (2017) 19.
[12] J.B. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability, 1, University of California Press, 1967, pp. 281–297.
[13] D.K. Tasoulis, D. Zeimpekis, E. Gallpoulos, M.N. Vrahatis, Oriented *k*-windows: a PCA driven clustering method, in: L. M, K. A., V. Z. (Eds.), Advances in Web Intelligence and Data Mining, Springer, 2006, pp. 319–329.
[14] P. Alevizos, D.K. Tasoulis, M.N. Vrahatis, Parallelizing the unsupervised *k*-windows clustering algorithm, in: Lecture Notes in Computer Science, 3019, 2004, pp. 225–232.
[15] G.S. Antzoulatos, E.K. Ikonomakis, M.N. Vrahatis, Efficient unsupervised clustering through intelligent optimization, in: IASTED 2009, Artificial Intelligence and Soft Computing (ASC), 2009, pp. 21–28.
[16] M. Pavan, M. Pelillo, Dominant sets and pairwise clustering, IEEE Trans. Pattern Anal. Mach. Intell. 29 (1) (2007) 167–172.
[17] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.
[18] G. Gan, M. Kwok-Po Ng, Subspace clustering using affinity propagation, Pattern Recognit. 48 (4) (2015) 1455–1464.
[19] X. Peng, Y. Wu, Large-scale cooperative co-evolution using niching-based multi-modal optimization and adaptive fast clustering", Swarm Evol. Comput. 35 (2017) 65–77.
[20] M. Gong, L. Jiao, L. Wang, L. Bo, Density-sensitive evolutionary clustering, in: 11th Pacific-Asia Conference in Advances in Knowledge Discovery and Data Mining, 38, 2007, pp. 507–514.
[21] J. de Andrade Silva, E.R. Hruschka, J.a. Gama, An evolutionary algorithm for clustering data streams with a variable number of clusters, Expert Syst. Appl. 67 (2017) 228–238.
[22] S. Paterlini, T. Krink, Differential evolution and particle swarm optimisation in partitional clustering, Computational Stat. Data Anal. 50 (2006) 1220–1247.
[23] S. Das, A. Abraham, A. Konar, Automatic clustering using an improved differential evolution algorithm, IEEE Trans. Syst. Man Cybern. 38 (2008) 218–237.
[24] L. Li, L. Jiao, J. Zhao, R. Shang, M. Gong, Quantum-behaved discrete multi-objective particle swarm optimization for complex network clustering, Pattern Recognit. 63 (2017) 1–14.
[25] R. Storn, K. Price, Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. 11 (4) (1997) 341–359.
[26] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, 4, 1995, pp. 1942–1948.
[27] D.E. Goldberg, Genetic algorithms in search, optimization and machine learning, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.
[28] J.H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, U Michigan Press, 1975.
[29] B.J. Frey, D. Dueck, Clustering by passing messages between data points, Science 315 (2007) 972–976.
[30] Z. Wei, Y. Wang, S. He, J. Bao, A novel intelligent method for bearing fault diagnosis based on affinity propagation clustering and adaptive feature selection, Knowl. Based Syst. 116 (2017) 1–12.
[31] L. Sun, C. Guo, C. Liu, H. Xiong, Fast affinity propagation clustering based on incomplete similarity matrix, Knowl. Inf. Syst. 51 (3) (2017) 941–963.
[32] L. Sun, C. Guo, Incremental affinity propagation clustering based on message passing, IEEE Trans. Knowl. Data Eng. 26 (11) (2014) 2731–2744.
[33] N.M. Arzeno, H. Vikalo, Semi-supervised affinity propagation with soft instance-level constraints, IEEE Trans. Pattern Anal. Mach. Intell. 37 (5) (2015) 1041–1052.
[34] F. Shang, L. Jian, J. Shi, F. Wang, M. Gong, Fast affinity propagation clustering: a multilevel approach, Pattern Recognit. 45 (1) (2012) 474–486.
[35] P. Li, H. Ji, B. Wang, Z. Huang, H. Li, Adjustable preference affinity propagation clustering, Pattern Recognit. Lett. 85 (2017) 72–78.
[36] A. Gionis, H. Mannila, P. Tsaparas, Clustering aggregation, ACM Trans. Knowl. Discov. Data 1 (1) (2007) 4.
[37] C.T. Zahn, Graph-theoretical methods for detecting and describing gestalt clusters, IEEE Trans. Comput. 20 (1) (1971) 68–86.
[38] H. Chang, D.-Y. Yeung, Robust path-based spectral clustering, Pattern Recognit. 41 (1) (2008) 191–203.
[39] L. Fu, E. Medico, Flame, a novel fuzzy clustering method for the analysis of dna microarray data, BMC Bioinform. 8 (1) (2007) 1.
[40] C.J. Veenman, M.J.T. Reinders, E. Backer, A maximum variance cluster algorithm, IEEE Trans. Pattern Anal. Mach. Intell. 24 (9) (2002) 1273–1280.
[41] R.A. Fisher, The use of multiple measurements in taxonomic problems, Ann. Eugen. 7 (2) (1936) 179–188.
[42] W.H. Wolberg, M.O. L., Multisurface method of pattern separation for medical diagnosis applied to breast cytology, in: Proceedings of the National Academy of Sciences, U.S.A., 87, 1990, pp. 9193–9196.
[43] M. Lichman, UCI Machine Learning Repository, 2013.
[44] P. Horton, K. Nakai, A probabilistic classification system for predicting the cellular localization sites of proteins, in: In Proceeding of the Fourth International Conference on Intelligent Systems for Molecular Biology, 1996, pp. 109–115.
[45] D.L. Davies, D.W. Bouldin, A cluster separation measure, IEEE Trans. Pattern Anal. Mach. Intell. 1 (2) (1979) 224–227.
[46] J.C. Dunn, Indices of Partition Fuzziness and the Detection of Clusters in Large Data Sets, Fuzzy Automata and Decision Processes, Elsevier, New York, 1977.
[47] P. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, J. Comput. Appl. Math. 20 (1987) 53–65.
[48] H. Xiong, Z. Li, Clustering validation measures, in: C.C. Aggarwal, C.K. Reddy (Eds.), Data Clustering, CRC Press, 2013, pp. 571–605.
[49] A. Rosenberg, J. Hirschberg, V-measure: a conditional entropy-based external cluster evaluation measure., in: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, 7, 2007, pp. 410–420.
[50] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning, Springer Series in Statistics, Springer New York, Inc., 2001.

Emmanouil K. Ikonomakis received his Master's degree on "Computational Intelligence" in 2009 from the Department of Mathematics, University of Patras. He is currently working towards his Ph.D. degree at the same department. He is the co-author of 6 peer reviewed publications and has over 260 citations. His research interests include Data Clustering, Medical Imaging and Evolutionary Optimization.

Dr. **George M. Spyrou** holds a BSc on Physics and MSc's on Medical Physics and Bioinformatics. During his PhD he worked on breast cancer imaging. He is the Bioinformatics ERA Chair Holder and the Head of the Bioinformatics Group at the

Cyprus Institute of Neurology and Genetics. He has taught courses and supervised MSc and PhD students in Postgraduate Programs at the University of Athens, at the Aristotle University of Thessaloniki and at the Democritus University of Thrace. His work includes computational methods that act as bridges between molecular biology, systems biology and molecular medicine, exploiting computational intelligence and high performance computing for multi-omics network analysis, systems bioinformatics and in silico drug discovery. He has authored over 160 scientific publications in peer reviewed journals and international conference proceedings.

Professor **Michael N. Vrahatis** obtained a Ph.D. in Mathematics in 1982 from the University of Patras, Greece. He is founder of the Computational Intelligence Laboratory of the same department which he has been directing since its beginning in 2004. He has been teaching at the undergraduate level for more than 33 years and at the post graduate level of more than 24 years. Several of his doctoral and/or postdoctoral students serve as full professors, associate professors, assistant professors or lecturers in Greece and England. He has published more than 400 scientific publications. According to Google Scholar his work has been cited over 14,000 times.