Applied Soft Computing 28 (2015) 69–80

Contents lists available at ScienceDirect

Applied Soft Computing

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

Dynamic clustering with improved binary artificial bee colony algorithm

Celal Ozturk*, Emrah Hancer, Dervis Karaboga

Erciyes University, Engineering Faculty, Computer Engineering Department, Kayseri, Turkey

ARTICLE INFO

Article history:
Received 16 July 2012
Received in revised form 26 June 2014
Accepted 30 November 2014
Available online 8 December 2014

Keywords:
Cluster analysis
Automatic clustering
Discrete optimization
Binary artificial bee colony algorithm

ABSTRACT

One of the most well-known binary (discrete) versions of the artificial bee colony algorithm is the similarity measure based discrete artificial bee colony, which was first proposed to deal with the uncapacited facility location (UFLP) problem. The discrete artificial bee colony simply depends on measuring the similarity between the binary vectors through Jaccard coefficient. Although it is accepted as one of the simple, novel and efficient binary variant of the artificial bee colony, the applied mechanism for generating new solutions concerning to the information of similarity between the solutions only consider one similarity case i.e. it does not handle all similarity cases. To cover this issue, new solution generation mechanism of the discrete artificial bee colony is enhanced using all similarity cases through the genetically inspired components. Furthermore, the superiority of the proposed algorithm is demonstrated by comparing it with the basic discrete artificial bee colony, binary particle swarm optimization, genetic algorithm in dynamic (automatic) clustering, in which the number of clusters is determined automatically i.e. it does not need to be specified in contrast to the classical techniques. Not only evolutionary computation based algorithms, but also classical approaches such as fuzzy C-means and K-means are employed to put forward the effectiveness of the proposed approach in clustering. The obtained results indicate that the discrete artificial bee colony with the enhanced solution generator component is able to reach more valuable solutions than the other algorithms in dynamic clustering, which is strongly accepted as one of the most difficult NP-hard problem by researchers.

1. Introduction

With the rapid development of computer hardware and software, datasets with great capacities can be stored without more effort. However, these datasets cannot be used or specified by users without of any pre-process. One of the new interdisciplinary fields of computer science, data mining concerns with datasets by basically trying to extract meaningful data and summarizing it into useful information through clustering, feature extraction, statistical tests, etc. [1]. In this study, general motivation just focuses on clustering, which is one of the most appreciated subjects by the researchers and is used in many real-world applications such as bioinformatics, machine learning, image analysis, and pattern recognition and market analysis. In clustering, the main goal is to divide data into groups or clusters based on some similarity measures like distance, intervals within multidimensional data [2]. Through clustering, valuable information can be extracted from enormous quantities of data.

Clustering algorithms fall into two main categories, hierarchical and partitional algorithms. Hierarchical algorithms are based on the use of proximity matrix indicating the similarity between every pair of data points to be clustered and its result is "dendrogram representing the nested grouping of patterns and similarity levels at which groupings change and levels are created through bottom up or bottom down approaches [2]". In agglomerative (bottom up) hierarchical algorithms, each data member is assigned to a unique cluster, then two clusters are found repeatedly according to the proximity matrix and finally they are merged. The basic agglomerative hierarchical algorithm has the following steps [3]: firstly, construct a similarity matrix which shows the difference between each pair of data. After that, assign $K = N$ where N is the number of data and K is the number of clusters. Then repeatedly find the nearest pair of distinct clusters, merge these clusters and decrement K, 1 by 1 while $K > 1$. The process of merging clusters can be applied by different ways, but the well-known are single link and complete link. Single link algorithms are based on merging two groups which have the smallest distance between their closest members. In contrast, complete link algorithms are based on merging groups which have the smallest distance between their most distant members. As for the divisive hierarchical algorithms, all data members are

* Corresponding author. Tel.: +90 352207666x32581; fax: +90 3524375784.
E-mail address: celal@erciyes.edu.tr (C. Ozturk).

http://dx.doi.org/10.1016/j.asoc.2014.11.040
1568-4946/

assigned to one cluster and then spitted a cluster at each stage until number of clusters is equal to the number of data points. Although the number of clusters is not predefined and the initial conditions do not affect the clustering process in hierarchical algorithms, these algorithms are not dynamic i.e. after a data point assigned to a cluster, its cluster cannot be updated, and the lack of information about global size and shape might cause overlapping clusters [4].

Contrary to hierarchical algorithms, partitional algorithms allow cluster members to be updated if it improves clustering performance. Partitional clustering attempts to decompose the data into a set of disjoint clusters using similarity criterion (e.g. square error function) which is tried to be minimized by assigning clusters to peaks in the probability density function, or the global structure [5]. Therefore, partitional clustering can be regarded as an optimization problem, which is also considered in this paper. In the usage of partitional clustering algorithms, the disadvantages of hierarchical algorithms are the advantages of partitional clustering algorithms, and vice versa [6].

Clustering can also be applied in two different modes: crisp (hard) and fuzzy (soft). Crisp clustering algorithms assume that each pattern should be assigned to only one cluster and the clusters are disjoint and non-overlapping. The most well-known example for the crisp clustering is the K-means algorithm. K-means [7] starts with K number of predefined clusters and then assigns each data member to its closest cluster. After the assignment, each cluster centroid is updated and this process is repeated until the termination criterion is satisfied. As in fuzzy clustering, a pattern may be assigned to all the clusters with a certain fuzzy membership function [2] (e.g. fuzzy C-means (FCM) [8]).

On account of the fact that K-means and FCM excessively depend on initial conditions, modifications have been proposed to improve the performance of the algorithms [9–12]. Moreover, evolutionary based clustering algorithms have been proposed in order to overcome local minima problem of these clustering approaches. Particle swarm optimization (PSO), proposed by Kennedy and Eberhart in 1995 [13,14], was applied to the clustering problems, and better performances were gained against K-means [15,16]. Omran and Al-Sharban [16] applied Baribones-PSO to image clustering problem. Wong et al. [17] proposed an improved version of the objective function, which was firstly proposed by Omran et al. [6]. Besides PSO, Hancer et al. [18,19] developed an artificial bee colony based brain tumour segmentation methodology from MRI images with the previously proposed objective function [6]. Ozturk et al. [20] determined the drawbacks of the objective functions in the literature and improved a new objective function satisfying well-separated and compact clusters. Moreover, the Ant colony optimization (ACO) was also applied to the clustering problem [21,22]. The detailed information can be found in [23–26].

It is clear that the number of clusters cannot be easily specified in many real world applications and datasets; therefore, the above mentioned algorithms requiring number of clusters as a parameter cannot be effectively employed. On behalf of these understanding, finding the "optimum" number of clusters in a data set has become an important research area. Proposed by Ball and Hall [27], ISODATA splits or merges clusters throughout the programme based on certain criteria in order to increase or decrease the number of clusters. However, ISODATA asks the user to specify the values of several parameters (e.g. the merging and splitting thresholds) and it can only merge two clusters under a user specified threshold [6]. Dynamic optimal cluster-seek (DYNOC) [28], similar to ISODATA, is based on maximizing the ratio of minimum inter-cluster distance to maximum intra cluster distance, but it also requires user specified parameters. Snob [29], Wallace's programme for unsupervised classification of multivariate data, uses the minimum [message or description] length [encoding] (MML or MMD) principle to decide upon the best classification of the data in order to assign objects to a cluster.

Evolutionary based algorithms have also been applied to the dynamic clustering problem, particularly in last decade. Omran et al. [4] proposed a PSO based dynamic image clustering (DCPSO), which was inspired by the ideas of Kuncheva and Bezdek [30]. In DCPSO, a cluster set (S) is first created and then binary PSO is applied to select cluster centroids from S. After that, the obtained cluster centroids from S within the best solution are refined by K-means. Das et al. [31,32] proposed differential evolution based algorithms (ACDE and AFDE) in which the parameters of F-scale and crossover rate are determined adaptively. In ACDE, each solution is represented by cluster centroids and associated activation values ([0,1]). Through evaluation, cluster centroids and their activators are updated simultaneously. Thus, it does not need to employ K-means to decrease the effects of initial conditions as in DCPSO. Kuo et al. [33] improved a hybrid PSO&GA algorithm to overcome the convergence problem of the PSO algorithm. However, the applied objective fitness function, based only on Euclidean distance, is not very convenient for dynamic clustering problem. Maulik and Saha [34] proposed a modified differential evolution clustering algorithm based on information of local and global best positions (MoDEAFC) to automatically extract information from remote sensing images. Rui et al. [35] employed DE and PSO sequentially for odd and even iterations and presented a comparative study on clustering validity indexes.

The main goal of this study is to demonstrate that the improved version of the discrete binary artificial colony algorithm (DisABC) [36] can be applied to the dynamic clustering problem. The novelty of the improved version of discrete artificial bee colony (referred as "IDisABC") comprises two parts: modified random selection and modified greedy selection. These improved selection mechanisms are applied to search solution space intimately with the help of crossover and swap operators when the number of probable obtained outputs ($M'vals$) by dissimilarity calculation of two neighbourhood solutions is greater than one. In this way, the computational complexity of the algorithm is not affected so much. Moreover, the performance analysis and performance comparisons of the algorithms have been tested on benchmark problems in terms of the index quality, obtained number of cluster and correct classification percentage (CCP) by applying the static algorithms such as K-means and FCM in addition to the evolutionary computation based algorithms, including the DCPSO, GA and DisABC. It should be noticed that CCP is one of the most significant criterions to measure the quality of clustering; however, not many studies, especially related to dynamic clustering, reported the values of CCP.

The rest of the paper is organized as follows; Section 2 describes the ABC algorithm; Section 3 demonstrates the IDisABC algorithm; Section 4 defines the clustering problem; and Section 5 presents the comparative results of the state of the art algorithms with the proposed algorithm. Finally, Section 6 concludes the paper.

2. Artificial bee colony algorithm

A model of intelligent behaviours of honey bee swarm introduced by Karaboga in 2005 [37], the artificial bee colony (ABC) algorithm is a novel swarm intelligence based algorithm and has been applied to various problems such as in optimization of numerical problems [38], data clustering [39], neural networks training for pattern recognition [40], wireless sensor network deployment [41] and routing [42] and image analysis [19,43]. The ABC algorithm comprises of three phases: employed bee phase, onlooker bee phase and scout bee phase. In employed bee and onlooker bee phases, a new solution is produced in the neighbourhood of current solution via Eq. (1);

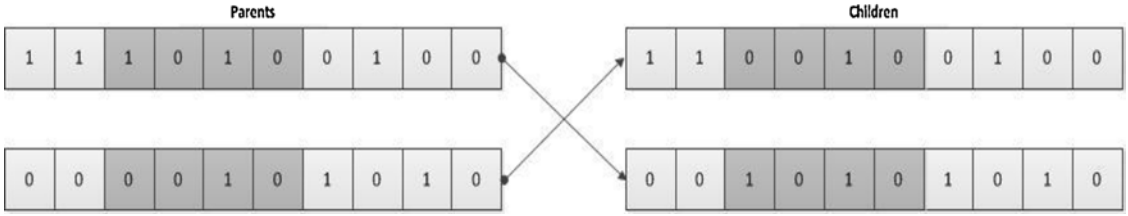

Fig. 1. An example of two point crossover.

$$V_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \tag{1}$$

where j and k are randomly chosen indexes and $j \neq k$, respectively, and ϕ_{ij} is a random number in [−1,1]. If the value of new solution V_i is better than the current one X_i, it is replaced with the current one. It should be noted that while employed bees search in the neighbourhood of current solutions, onlookers search in the neighbourhood of the solutions selected by applying probabilistic manner via Eq. (2) and (3). By this way, more qualified solutions are chosen to be improved by the bees;

$$fitness_i = \begin{cases} \dfrac{1}{1+fit_i}, & fit_i \geq 0 \\ 1 + abs(fit_i), & fit_i < 0 \end{cases} \tag{2}$$

$$p_i = \frac{fitness_i}{\sum_{SN}^{j=1} fitness_i} \tag{3}$$

where fit_i is the cost value (value of objective function) of the food source X_i and SN is the number of solutions.

After the processes of the employed and onlooker bee phases, scout bee phase checks whether there exists any abandoned food source, which exceeds a predetermined number of trials called "limit". If so, a new solution is generated by Eq. (4) instead of the abandoned one;

$$x_{ij} = x_j^{min} + rand\ (0,1)(x_j^{max} - x_j^{min}) \tag{4}$$

where $i = 1, \ldots, SN$, SN is the number of solutions, $j = 1, \ldots, D$, D is the number of parameters. x_j^{min} is the minimum and x_j^{max} is the maximum values of parameter j. More information on bee swarm intelligence and artificial bee colony algorithm can be found in [44,45].

3. Improved binary artificial bee colony algorithm (IDisABC)

The basic ABC is not suitable to the binary optimization problems since ABC was first proposed for the optimization of numerical problems. To apply ABC in binary problems, Pampapa and Engelbrecht [46] improved an angle modulation based binary ABC model (AMABC), in which each solution is represented by four dimensional real tuple and is then transformed into binary form for the evaluation of objective function. Kashan et al. [36] proposed a concept of dissimilarity between the binary vectors as a measure to evaluate how far the two binary vectors are apart from each other and used that concept to construct the binary ABC model (DisABC). Kiran and Gunduz [47] integrated a simple XOR operator based neighbourhood search mechanism to the ABC algorithm to solve the uncapacited facility location (UFLP) problem. Wei et al. [48,49] used two neighbourhoods (as in DE) and round operator to generate binary solutions. To our knowledge, not so many studies on binary ABC models can be reached in the literature, which motivates us to develop a new one.

In this paper, the DisABC algorithm is modified by two improved selection schemes through genetic operators in the phase of new solution generation (referred as IDisABC). The applied operators in modified selection schemes are as follows:

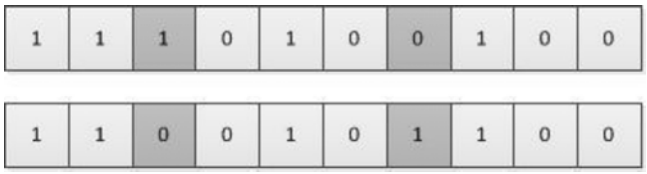

Fig. 2. An example of swap operator.

(1) *Crossover operator*: Crossover is used in population in order to increase possibility of getting optimal solution. The most widely used crossover operators are one-point, two-point and uniform. In this study, two-point is chosen as a crossover operator, where two positions on binary vectors are randomly selected and then the positions of the vectors between the determined positions are exchanged between binary vectors. An example of applying two-point crossover can be illustrated in Fig. 1.
(2) *Swap operator*: Swap is the act of exchanging information between objects in such a way that two positions, the values of which are 0 and 1, are switched to each other i.e. total number of ones and zeros in the binary vector is not changed, an example of which is presented in Fig. 2.

To generate a solution in the neighbourhood of current solution in IDisABC, similarity measure is first calculated between them as in DisABC;

(A) *Similarity measure*: The similarity between two vectors provides information about how far apart they are from each other and how similar they are to each other. Jaccard's coefficient is used to quantify the degree of similarity.

Let $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, $X_k = (x_{k1}, x_{k2}, \ldots, x_{kD})$ are two binary vectors where D is the dimension of vectors. The similarity between X_i and X_k is defined by Eqs. (5) and (6);

$$Similarity\ (X_i, X_k) = \frac{M_{11}}{M_{01} + M_{10} + M_{11}} \tag{5}$$

$$Dissimilarity\ (X_i, X_k) = 1 - Similarity\ (X_i, X_k) \tag{6}$$

where M_{11} is the total number of bits where both X_i and X_k are equal to 1, M_{01} is the total number of bits where X_i is equal to 0 and X_k is equal to 1 and M_{10} is the total number of bits where X_i is equal to 1 and X_k is equal to 0.

After calculation of the similarity, *M'vals* are calculated;

(B) *Calculation of M'vals*: Eq. (1) can be defined in the form of $V_i - x_i = \phi(x_i - x_k)$. The dissimilarity measure is placed with "−" operator. Then, the new differential equation can be defined by Eq. (7);

$$Dissimilarity\,(V_i, x_i) \approx \phi Dissimilarity\,(x_i, x_k) \tag{7}$$

Let m_1 is the number of bits with value 1 in X_i and m_0 is the number of bits with value 0 in X_i.

$$\min \left| 1 - \frac{M_{11}}{M_{01} + M_{10} + M_{11}} - \phi Dissimilarity\,(x_i, x_k) \right| \tag{8}$$

$$M_{11} + M_{01} = m_1 \tag{9}$$

$$M_{10} \leq m_0 \tag{10}$$

$$M_{11}, M_{10}, M_{01} \geq 0 \text{ and they are integer} \tag{11}$$

Eqs. (8)–(11) are used to produce new combinations with the values of M_{11}, M_{10}, M_{01} which will be the total number of bits where both X_i and V_i are equal to 1 (M_{11}), the total number of bits where X_i is equal to 0 and V_i is equal to 1 (M_{10}) and the total number of bits where X_i is equal to 1 and V_i is equal to 0 (M_{01}). Eq. (8) tries to minimize the gap between X_i and V_i. Eq. (9) satisfies the maximum number of combinations of M_{11}, M_{10}, M_{01} values, that is $(m_1 + 1)(m_0 + 1)$.

It should be notified that it is possible to get more than one *M'vals* combinations satisfying Eq. (8), which is not considered by DisABC i.e. DisABC only gets the first M_{11}, M_{10}, M_{01} combination value to generate solution. However, IDisABC considers all possible *M'vals* combinations through the following modified selection mechanisms in a probabilistic manner;

1. *Modified random selection*: Firstly, V_i is initialized with zeros. If there exists only one optimal value for tuple M_{11}, M_{10} and M_{01}, M_{11} number of positions with value one are randomly selected from X_i and these positions in V_i are assigned as one. After that, the M_{10} number of positions with value zero is randomly selected from X_i and these positions in V_i are also assigned as one. On the other hand, if the number of M_{11}, M_{10} and M_{01} combinations is more than 1, a new solution is generated (referred as MV_i's solution vector) for each tuple M_{11}, M_{10} and M_{01}, as firstly mentioned in random selection. Then, two point crossover operator is applied to the X_i, X_{global} and MV_i's, and then swap operator is applied to the generated solutions obtained by crossover. Finally, the best solution is chosen as V_i.
2. *Modified greedy selection*: Firstly, V_i is initialized with zeros. If there exists only one optimal value for tuple M_{11}, M_{10} and M_{01}, the positions with value one at both X_i and X_{global} are selected and these positions in V_i are assigned as one. After the assignment, if the number of changed ones (referred as $index_0$) is less than M_{11}, $M_{11} - index_0$ number of positions with value one are chosen from X_i and these positions in V_i are updated according to the random selection. After that, all positions which are zero at both X_i and X_{global} are also assigned as one in V_i. If the number of changed zeros (referred as $index_1$) is less than M_{10}, $M_{10} - index_1$ number of positions with value zero are selected from X_i and these positions in V_i are updated according to random selection. On the other hand, if the number of M_{11}, M_{10} and M_{01} combinations is more than 1, a new solution V_i is generated for every tuple M_{11}, M_{10} and M_{01} as firstly mentioned in greedy selection. After that, two point crossover operator is applied among X_i, X_{global} and generated solutions of MV_i's, and then swap operator is applied to the generated solutions obtained by crossover. Finally, the best solution is chosen as V_i.

To show the difference between IDisABC and DisABC in generating new solution V_i, the same example is used as in [36]. Let consider two vectors;

$X_i = (1011010100)$ and $X_k = (1000101101)$.

Assume that $\phi = 0.7$; $M_{11} = 2$, $M_{10} = 3$ and $M_{01} = 3$ are founded from X_i and X_k.

$$\phi Dissimilarity\,(x_i, x_k) = 0.7 \left(1 - \frac{2}{2+3+3}\right) = 0.525$$

m_0 and $m_1 = 5$ from X_i. Then by Eqs. (8)–(11);

$$\min z = \left| 1 - \frac{M_{11}}{M_{01} + M_{10} + M_{11}} - 0.525 \right|$$

$$M_{11} + M_{01} = 5$$

$$M_{10} \leq 5$$

$$M_{11}, M_{10}, M_{01} \geq 0 \quad \text{and integer}$$

In DisABC, the optimal output is $M_{11} = 3$, $M_{01} = 2$, $M_{10} = 1$ and $z = 0.025$. V_i is initialized with 1x10 zeros. $M_{11} = 3$ number of positions are randomly selected from X_i and these positions are set to 1 in V_i. Then, $M_{10} = 1$ number of positions is randomly selected from X_i and it is assigned as 1 in V_i. The final generated solution is $V_i = (1001000110)$.

In IDisABC, all the optimal outputs are selected according to $z = 0.025$:

- $M_{11} = 3$, $M_{01} = 2$ and $M_{10} = 1$;
- $M_{11} = 4$, $M_{01} = 1$ and $M_{10} = 3$;
- $M_{11} = 5$, $M_{01} = 0$ and $M_{10} = 5$;

For every group of M values, a new solution is generated as in DisABC (referred as MV_i's solutions). Two point crossover is applied among all X_i, X_{global} and MV_i's solutions. After that, swap operator is applied to the solutions generated by two-point crossover. Finally, the best solution within all generated solutions is assigned as V_i. Furthermore, the implementation differences between the DisABC and IDisABC algorithms can be also clearly illustrated in pseudo codes, shown in Figs. 3 and 4.

4. The clustering problem

Patterns are fundamental structures of an object used by clustering and are distinguished from each other through their attributes known as features. Given a data set $Z = \{z_1, z_2, \ldots, z_p, \ldots, z_N\}$, where z_p is a pattern with N_d dimensional feature space and N is the number of patterns in Z. When Z is demanded for partitioning into K clusters as $C = \{C_1, C_2, \ldots, C_k\}$ through a crisp clustering approach, the following rules needs to be considered:

- Each cluster should have at least one pattern assigned, $C_i \neq \emptyset \; \forall \, i = 1, 2, \ldots, K$
- Each pattern should be assigned to a cluster, $\bigcup_{k=1}^{K} C_k = Z$
- Each pattern should be assigned to only one cluster, $C_i \cap C_j = \phi$ where $i \neq j$

In addition to these rules, measuring similarity between the patterns should be also handled in clustering, where distance measure, especially Euclidean distance given by Eq. (12), is most widely used to evaluate the similarity of patterns [15]:

$$d(z_v, z_y) = \sqrt{\sum_{j=1}^{N_d} (z_{v,j} - z_{y,j})^2} = \| z_v - z_y \|_2 \tag{12}$$

Moreover, how to evaluate the performances of clustering algorithms and how to determine optimal number of clusters are the other significant subjects in clustering. To measure the quality of clustering relative to others created by other methods or by the same methods using different parameter values and to determine the number of clusters, clustering validity indexes are used. It can be clearly stated that various validity indexes in the literature, including the Dunn's index (DI) [50], Calinski-Harabasz index [51], DB index [52], CS measure [53], VI index [3], etc. have been proposed

```
begin
    Initialize food sources;
    Find the global best (gbest) food source;
    for cycle ← 1 to MCN do
        foreach employed bee i do
            Choose a food source X_k in the neighbourhood of X_i;
            Calculate Dissimilarity(X_i, X_k) by Eqs.5-6;
            Find one M'val combination by Eqs. 8-11;
            if rand() < 0.5 then
                V_i = randomselection(Mvals, X_i);
            else
                V_i = greedyselection(Mvals, X_i);
            end
        end
        foreach onlooker bee i do
            Select a food source X_i depending on probability p_i using Eq.3;
            Choose a food source X_k in the neighbourhood of X_i;
            Calculate Dissimilarity(X_i, X_k) by Eqs.5-6;
            Find one M'val combination by Eqs. 8-11;
            if rand() < 0.5 then
                V_i = randomselection(Mvals, X_i);
            else
                V_i = greedyselection(Mvals, X_i);
            end
        end
        Apply local search module;
        if there exits an abondoned food source then
            Scout bee determines a new food source by Eq.4;
        end
        Update gbest food source;
    end
end
```

Fig. 3. The pseudo code of the DisABC algorithm.

and all of them, despite some differences, should satisfy the following properties [6,54]:

Compactness: Within the cluster, patterns should close (similar) to each other as much as possible.

Separation: Clusters should be well separated from each other in such a way that cluster centroids are distant from each other as much as possible.

In this study, the VI index is chosen as an objective function on account of its satisfactory performance [3,6]. The components of the VI index are as follows:

(1) *Intra-cluster distance*: The average of all distances between patterns and their related centroids, intra-cluster distance is used to measure compactness. In order to satisfy well compact clusters, intra-cluster distance should be minimized, defined by Eq. (13):

$$\text{intra} = \frac{1}{N_p}\sum_{k=1}^{K}\sum_{z\in C_k} \| z_p - m_k\|^2 \tag{13}$$

where N_p is the number of patterns in a dataset, z_p is a pattern in cluster C_k, m_k is the kth cluster centroid and K is the number of clusters.

(2) *Inter-cluster distance*: Known as minimum Euclidean distance between any pairs of cluster centroids, inter-cluster distance is used to measure the separateness of clusters. It should be maximized to obtain well separated clusters, defined by Eq. (14):

$$\text{inter} = \min\{\| m_k - m_{kk}\|^2\} \quad \text{where} k \neq kk \tag{14}$$

With the knowledge of the intra and inter cluster distances, the VI index can be defined by Eq. (15):

$$VI = (c \times N(\mu, \sigma) + 1) \times \frac{\text{intra}}{\text{inter}} \tag{15}$$

where c is a user specified constant (mostly chosen as 20, 25 or 30) and $N(\mu,\sigma)$ is a Gaussian distribution with mean μ and standard deviation σ, defined by Eq. (16):

$$N(\mu, \sigma) = \frac{1}{\sqrt{2\Pi\sigma^2}} e^{[(K-\mu)^2/2\sigma^2]} \tag{16}$$

where K is the number of clusters. Turi [3] tried to solve problems including more than 4 clusters; hence, μ was limited to 2. In addition to the higher number of clusters, this paper also concentrates on problems regarding fewer clusters (less than 4); therefore, μ is taken 0 or 1 for fewer clusters and taken 2 or 3 for higher clusters, and σ is assigned as 1 for all cases.

It should be noted that when assigning patterns to the clusters, a cluster may become empty or there exist only one cluster. To sort out this problem, new solution is generated until the obtained number of clusters is more than one and there exist no empty cluster. The implementation steps of an evolutionary based algorithm to the dynamic clustering are given as follows:

1. Initialize randomly N_c cluster centroids;
2. Generate the initial solutions; X_i: $i = 1 \ldots SN$ by Eq. (17);

$$x_{ij} = \begin{cases} 0 & \text{if} U(0,1) \leq 0.5 \\ 1 & \text{if} U(0,1) > 0.5 \end{cases} \tag{17}$$

```
begin
    Initialize food sources;
    Find the global best (gbest) food source;
    for cycle ← 1 to MCN do
        foreach employed bee i do
            Choose a food source X_k in the neighbourhood of X_i;
            Calculate Dissimilarity(X_i, X_k) by Eqs.5-6;
            Find the M'vals combinations by Eqs. 8-11;
            if rand() < 0.5 then
                if the number of M'vals combinations==1 then
                    V_i = randomselection(Mvals, X_i);
                else
                    foreach Mvals k do
                        MV_k = randomselection(Mvals_k, X_i);
                    end
                    Apply two-point crossover between gbest, X_i and MV's sources;
                    Apply swap operator to the food sources generated by two-point crossover;
                    Select the best food source as V_i among the generated food sources;
                end
            else
                if the number of M'vals combinations==1 then
                    V_i = greedyselection(Mvals, X_i);
                else
                    foreach Mvals k do
                        MVl_k = greedyselection(Mvals_k, X_i);
                    end
                    Apply two-point crossover between gbest, X_i and MV's sources;
                    Apply swap operator to the food sources generated by two-point crossover;
                    Select the best food source as V_i among the generated food sources;
                end
            end
        end
        foreach onlooker bee i do
            Select a food source X_i depending on probability p_i using Eq.3;
            Choose a food source X_k in the neighbourhood of X_i;
            Calculate Dissimilarity(X_i, X_k) by Eqs.5-6;
            Find the M'vals combinations by Eqs. 8-11;
            if rand() < 0.5 then
                if the number of M'vals combinations==1 then
                    V_i = randomselection(Mvals, X_i);
                else
                    foreach Mvals k do
                        MV_k = randomselection(Mvals_k, X_i);
                    end
                    Apply two-point crossover between gbest, X_i and MV's sources;
                    Apply swap operator to the food sources generated by two-point crossover;
                    Select the best food source as V_i among the generated food sources;
                end
            else
                if the number of M'vals combinations==1 then
                    V_i = greedyselection(Mvals, X_i);
                else
                    foreach Mvals k do
                        MV_k = greedyselection(Mvals_k, X_i);
                    end
                    Apply two-point crossover between gbest, X_i and MV's sources;
                    Apply swap operator to the food sources generated by two-point crossover;
                    Select the best food source as V_i among the generated food sources;
                end
            end
        end
        if there exits an abondoned food source then
            Scout bee determines a new food source by Eq.4;
        end
        Update gbest food source;
    end
end
```

Fig. 4. The pseudo code of the IDisABC algorithm.

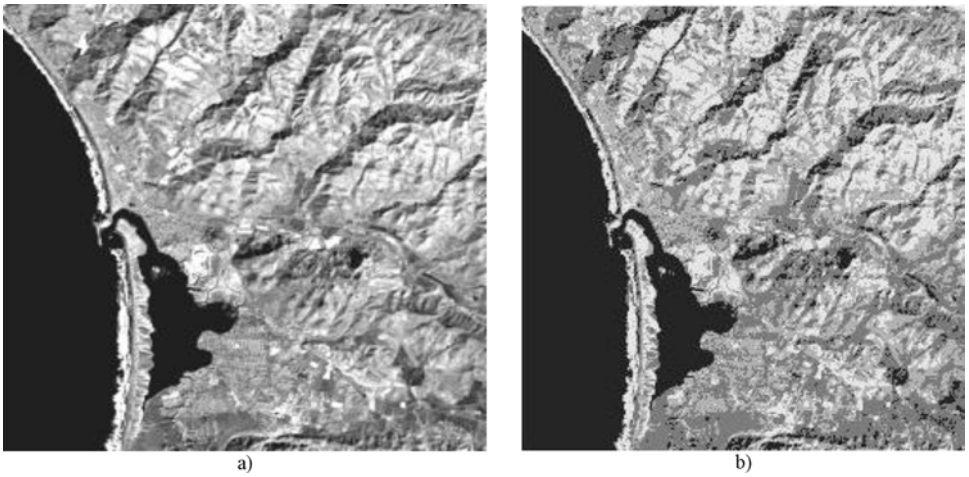

Fig. 5. (a) Original Morro Bay Image; (b) clustered by IDisABC into 4 clusters.

3. Set cycle to 1
4. For each solution X_i
   a. Activate the centroids regarding to the dimensions the value of which are 1;
   b. Calculate the Euclidean distances of each pattern from the activated centroids by Eq. (12);
   c. Assign each pattern to its closest cluster;
   d. Calculate $f(X_i)$ = VI by Eq. (15);
5. Update solutions by employed evolutionary computation based algorithm;
6. Repeat (3) to (5) until cycle = MCN
7. Apply K means to the remained centroids
8. Generate a new cluster set as in step 1
9. Merge new cluster set with remained centroids
10. Repeat steps (2)–(9) until termination criterion (TC) is satisfied.

5. Experimental results

In the experimental study, the IDisABC based dynamic clustering algorithm is applied to the fields of grayscale image and data clustering, and its superiority is indicated by comparing it with the DisABC, DCPSO and GA based clustering algorithms. In addition to the automatic clustering algorithms, the most well-known static algorithms where the number of clusters is predefined by a user, K-means and FCM are also employed to make a comprehensive analysis. Simulation of each algorithm is performed for 30 times and the mean and std values are presented in tables, where the standard deviation values of produced results are reported in parenthesis and the best results are denoted in bold. In the tables, the results of validity index are reported to measure the quality of clustering and to show the success of compared algorithms, and the obtained number of clusters is presented to demonstrate the performance of the algorithms in partitioning process. Furthermore, the correct classification percentage (CCP) values are presented additional to the validity index and obtained cluster numbers in data clustering;

$$CCP = 100 \times \frac{\text{number of correctly classified data}}{\text{total number of data set}} \quad (18)$$

In the experiments of grayscale image clustering, the following control parameter values are used for the DisABC, IDisABC, DCPSO and GA algorithms; the number of population is chosen as 50, the maximum number of elevations is set to 10,000, and the termination criterion (TC) is assigned as 2. As for the data clustering, the number of population and the maximum number of elevations are selected as 30 and 9000, respectively, and TC is assigned as 3. The other control values of the evolutionary computation based algorithms are chosen as follows: in DisABC and IDisABC the limit value is set to 100 and 20; in DCPSO $c_1 = 2$, $c_2 = 2$, $w_{start} = 0.9$, $w_{end} = 0.4$ and $V_{max} = 6$; and in GA crossover rate and mutation rate are selected as 0.8 and 0.01, respectively.

Five benchmark grayscale images, including Morro Bay, Lena, Mandril, Jet and Pepper (shown in Figs. 5–9) are chosen to test the performances of the algorithms in image clustering and five well-known benchmark data sets, including Wisconsin, Iris, Wine, Ecoli and Dermatology from the UCI Machine Learning Repository are chosen for data clustering. The first applied data set, Wisconsin breast cancer consists of 683 data members having 9 features and 2 classes used to group benign tumours and malignant tumours. The second data set introduced by Sir Ronald Fisher in 1936, Iris consists of 3 classes, 4 features and 150 data members. The aim is to classify each data member into Setosa, Virginica or Versicolour. The other applied data sets, Wine determines three types of wines consisting of 13 features and 178 data members, Ecoli consists of 5 classes, 8 features and 327 data members and Dermatology consists of 6 classes (psoriasis, seboreic dermatisis, lichen planus, pityriasis rosea, cronic dermatitis and pityriasis rubra pilaris), 34 features and 358 data members. It should be noted that data sets are used in normalized form.

In the experiments, the following parameter values are chosen for VI index; c is set to 25 and 30 for image clustering [6] and data clustering [33], respectively. For the images Lena, Mandril and Jet; μ is chosen 2 and for Pepper, μ is chosen 3. Since the target number of clusters is below than 5 for Morro Bay, μ is chosen 1. In data clustering, for Ecoli and Dermatology, μ is taken 2 and for Iris, Wine and Wisconsin, μ is taken 0 [33]. The maximum number of clusters (N_c) is selected according to the target number of clusters; 13 for Wisconsin, 15 for Iris and Wine, 20 for Dermatology and Ecoli, 15 for Morro Bay and 20 for the other images.

Evaluating the performances of the considered evolutionary computation based algorithms in image sets, Table 1 shows that the IDisABC based clustering algorithm is superior to the other clustering algorithms in all cases in terms of minimizing the VI index value. Moreover, the standard deviation values of the IDisABC algorithm are the lowest values for all images in Table 1, which indicates the robustness of the proposed algorithm is also satisfactory. When considering the other algorithms in terms of VI index, it is clearly seen from Table 1 that GA is not able to minimize the VI index in all cases and BPSO is not able to minimize the VI index well in Jet and Pepper images. Therefore, the algorithms such as DisABC, DCPSO and GA are ranked as second, third and last positions. In terms of obtaining the optimal number of clusters, Table 2 demonstrates that the IDisABC algorithm performs better than the others

Fig. 6. (a) Original Lena Image; (b) clustered by IDisABC into 4 clusters.

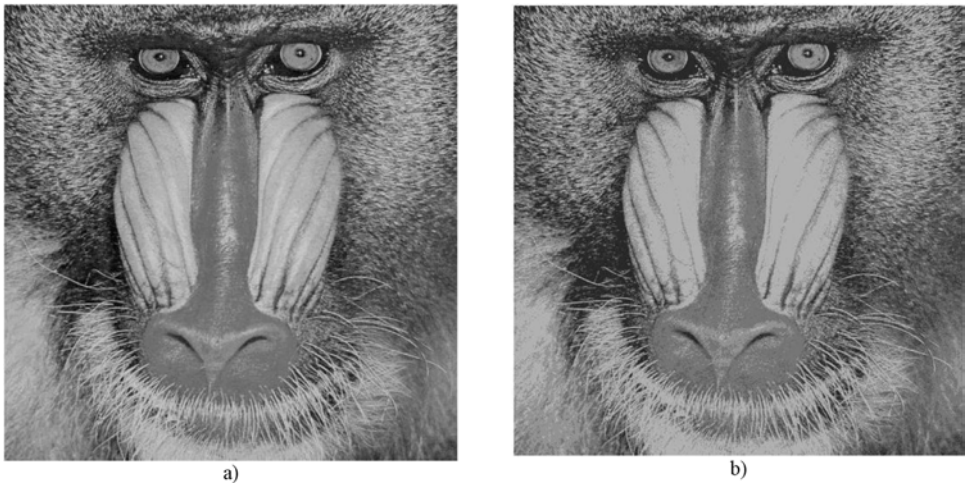

Fig. 7. (a) Original Mandril Image; (b) clustered by IDisABC into 6 clusters.

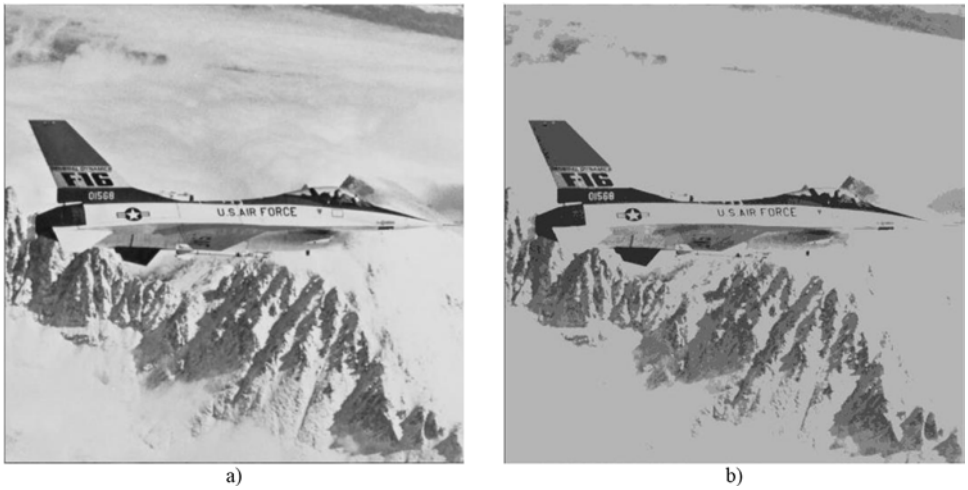

Fig. 8. (a) Original Jet Image; (b) clustered by IDisABC into 6 clusters.

Fig. 9. (a) Original Pepper Image; (b) clustered by IDisABC into 7 clusters.

Table 1
The results of VI Index for image clustering.

Images	IDisABC	DisABC	DCPSO	GA
MorroBay	**0.0789** (0.0093)	0.0875 (0.0125)	0.1053 (0.0339)	0.1268(0.0395)
Lena	**0.0982** (0.0118)	0.1032 (0.0141)	0.1126 (0.0130)	0.1395 (0.0259)
Mandril	**0.1043** (0.0109)	0.1045 (0.0111)	0.1077 (0.0115)	0.1419 (0.0324)
Jet	**0.0922** (0.0209)	0.0959 (0.0305)	0.1366 (0.1183)	0.1517 (0.0595)
Pepper	**0.1081** (0.0142)	0.1201 (0.0150)	0.1323 (0.0460)	0.1662 (0.0595)

Table 2
The obtained number of clusters for image clustering.

Images	#clusters	IDisABC	DisABC	DCPSO	GA
MorroBay	4	**4.333** (0.479)	**4.333** (0.479)	**4.333** (0.546)	4.621 (1.146)
Lena	6	**5.9** (0.922)	5.666 (0.546)	6.695 (1.063)	7.1 (1.516)
Mandril	6	**5.966** (0.905)	6.166 (1.0199)	6.9 (1.471)	7.4 (1.652)
Jet	6	5.733 (0.868)	5.5 (0.861)	**6.033** (0.964)	6.733 (1.680)
Pepper	7	**6.733** (0.691)	6.566 (0.568)	7.333 (1.212)	7.466 (1.105)

almost in all cases except Jet image. The best performance for Jet is achieved by DCPSO and all employed evolutionary based algorithms except GA obtains the same results in Tahoe image. Thus, the other algorithms such as DisABC, DCPSO and GA are ranked as second, third and last regarding to their performances in most cases and it is indicated from Table 2 that GA is not suitable to the image clustering. For instance, GA could only get 7.1 and 7.4 mean values for the images Lean and Mandril, which is not applicable when compared to the others. Thus, Tables 1 and 2 conform to the fact that IDisABC algorithm achieves the best clustering results regarding both index quality and optimal number of clusters. As for the comparison of the proposed method with the well-known static approaches in terms of VI index, Table 6 clearly shows that both of the K-means and FCM algorithms cannot satisfy sufficient

Table 3
The results of VI Index for data clustering.

Data set	IDisABC	DisABC	DCPSO	GA
Wisconsin	**0.135** (0.022)	**0.135** (0.022)	0.1368 (0.018)	0.1424 (0.039)
Iris	**0.0974** (0.010)	0.0982 (0.010)	0.1042 (0.013)	0.1182 (0.020)
Wine	**0.3251** (0.035)	0.3365 (0.029)	0.3518 (0.055)	0.4426 (0.145)
Ecoli	**0.3073** (0.057)	0.3841 (0.069)	0.3691 (0.086)	0.5351 (0.176)
Dermatology	**0.3968** (0.043)	0.4328 (0.048)	0.4600 (0.045)	0.5717 (0.138)

Table 4
The obtained number of clusters for data clustering.

Data set	#clusters	IDisABC	DisABC	DCPSO	GA
Wisconsin	2	**2.033** (0.018)	**2.033** (0.018)	2.066 (0.253)	2.133 (0.681)
Iris	3	**3** (0)	**3** (0)	**3** (0)	3 (0)
Wine	3	**3.3** (0.534)	3.4 (0.498)	3.9 (0.712)	4.3 (1.557)
Ecoli	5	**5.166** (0.379)	5.333 (0.606)	5.866 (0.776)	7.7 (1.600)
Dermatology	6	5.56 (0.504)	5.533 (0.571)	**6.033** (0.999)	6.966 (1.790)

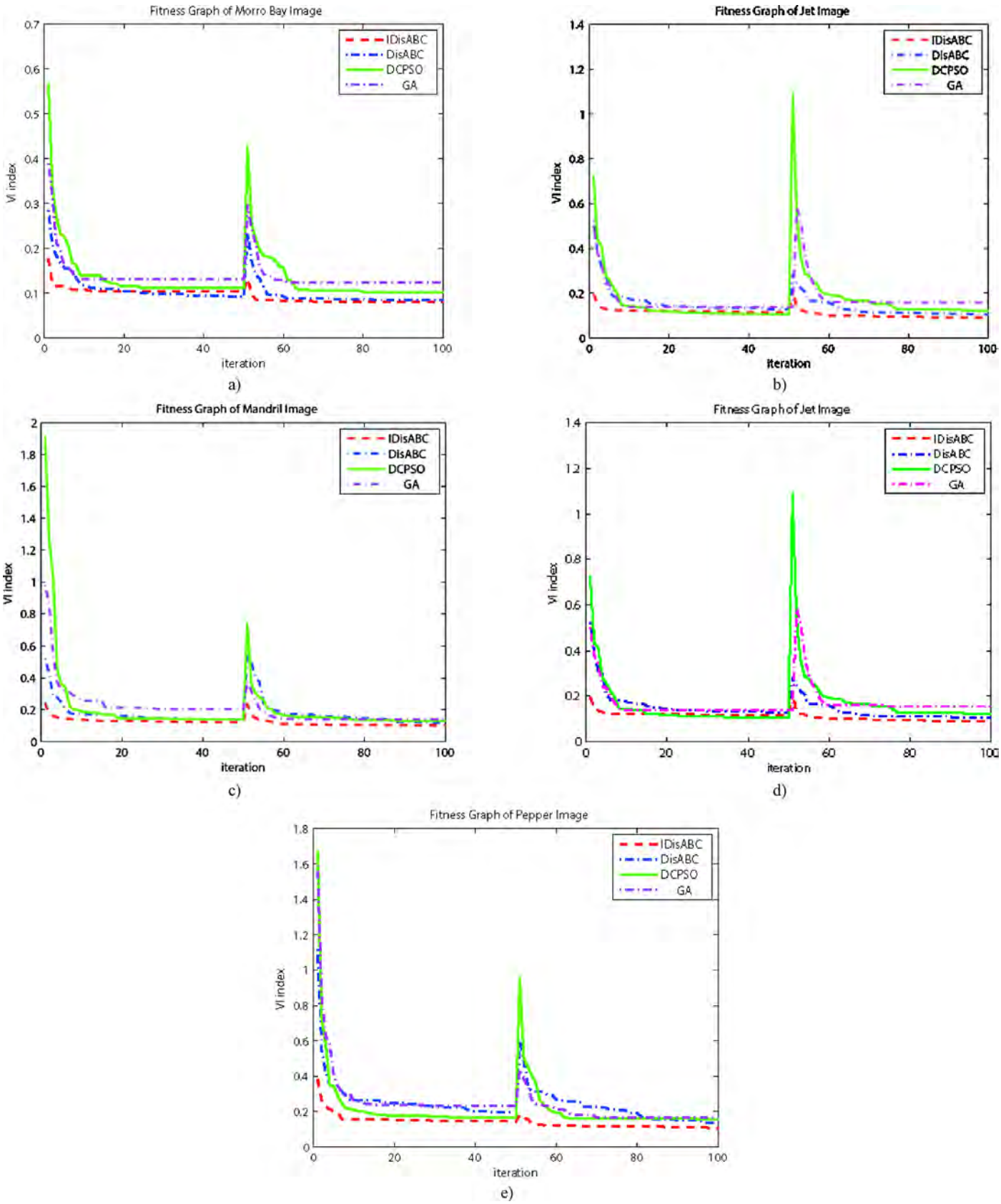

Fig. 10. Convergence graphics of IDisABC, DisABC, DCPSO and GA for the images.

Table 5
The results of CCP for data clustering.

Data Set	IDisABC	DisABC	DCPSO	GA
Wisconsin	**95.412** (0.361)	**95.412** (0.361)	95.314 (0.390)	92.967 (7.311)
Iris	**96** (0)	**96** (0)	**96** (0)	94.044 (7.442)
Wine	**91.928** (1.184)	91.835 (1.196)	89.189 (4.049)	71.647 (20.647)
Ecoli	**80.601** (2.445)	79.816 (2.471)	78.848 (4.723)	49.928 (15.177)
Dermatology	**79.935** (6.787)	78.873 (7.111)	77.357 (8.239)	68.286 (14.439)

Table 6
The VI Index results of Idisabc, K-means and FCM algorithms for image clustering.

Images	IDisABC	K-means	FCM
MorroBay	**0.0789** (0.0093)	1.6597 (0.0766)	1.4506 (8.65E−05)
Lena	**0.0982** (0.0118)	0.3215 (0.0238)	0.2863 (0.0002)
Mandril	**0.1067** (0.0144)	0.1371 (0.0053)	0.1250 (0.0002)
Jet	**0.0982** (0.0142)	0.7698 (0.0064)	0.7194 (1.16E−06)
Pepper	**0.1081** (0.0142)	0.6488 (0.2239)	0.4432 (0.0015)

Table 7
The results of Idisabc, K-means and FCM algorithms for data clustering.

Data Set	IDisABC		K-means		FCM	
	VI Index	CCP	VI Index	CCP	VI Index	CCP
Wisconsin	**0.135** (0.022)	95.412 (0.361)	0.178 (0.001)	**95.461** (0.287)	0.1754 (1.99E−05)	94.729 (0)
Iris	**0.0974** (0.010)	**96** (0)	0.8039 (1.127)	86.955 (18.520)	0.135 (0.001)	**96** (0)
Wine	**0.3251** (0.035)	91.928 (1.184)	0.5623 (0.003)	**92.696** (4.34E−14)	0.7307 (0.001)	92.546 (0.252)
Ecoli	**0.3073** (0.057)	**80.601** (2.445)	0.9553 (0.202)	57.043 (14.860)	2.9879 (0.152)	58.114 (3.109)
Dermatology	**0.3968** (0.043)	**79.935** (6.787)	1.7953 (0.826)	63.147 (16.176)	>5	25.363 (23.674)

performances. To observe the results visually, the original and clustered images are demonstrated in Figs. 5–9 for each benchmark image. In addition, convergence graphics of evolutionary based algorithms are drawn to analyze the results of image clustering in Fig. 10. As seen in Fig. 10, the convergence graphics of all algorithms include peaks since the algorithms run two times for each simulation i.e. TC is set to 2. It can be also clearly extracted from Fig. 10 that since the peaks of the IDisABC algorithm are less than the others, it is the most successful algorithm in terms of convergence rate. Therefore, it can be noticed that the IDisABC algorithm is the most robust algorithm.

Concerning to data clustering, Tables 3–5 demonstrate the results of VI index, optimal number of clusters and CCP. It can be inferred from Table 3 that IDisABC gets the best performances in minimizing the VI index in all cases. When comparing with the others, it is seen that there exist great differences between the IDisABC and the others in VI index quality. For instance, while IDisABC gets only 0.30 in Ecoli, the others such as DisABC, DCPSO and GA gets 0.38, 036 and 0.54, respectively. As for the comparison of others, DisABC performs better than GA and BPSO, and GA also gets the worst performance as in the image clustering. In terms of obtained number of clusters, IDisABC, DisABC and DCPSO mostly achieves optimum number of clusters in Wisconsin and Iris. For the other cases, IDisABC gets the best obtained number of clusters except Dermatology. Although DCPSO shows better performance for dermatology in terms of finding optimal number of clusters, its standard deviation value is much higher than the IDisABC and DisABC algorithms. Like as the image clustering, GA unfortunately cannot get well obtained number of clusters in data clustering. In terms of the CCP presented in Table 5, while the DCPSO and IDisABC algorithms obtain the best values for 1 and 2 cases, the IDisABC algorithm provides the best performances in all cases; on the other hand, GA gets the worst CCP values. Therefore, it can be indicated that the IDisABC algorithm also outperforms the other evolutionary computation based algorithms in data clustering. Presenting the obtained results of the proposed algorithm with the statistical approaches, Table 7 points out the obtained results of K-means and FCM cannot be accepted as sufficient concerning the VI index. Moreover, they only outperform the proposed algorithm for only 2 cases in terms of the CCP values.

6. Conclusion

The main goal is to improve new solution generation mechanism of the DisABC algorithm for dynamic clustering. This goal is achieved by considering all similarity cases through genetic components in an efficient way. In detailed, a solution is generated according to each similarity case and then crossover and swap are applied among them. The performance analysis of the proposed algorithm (IDisABC) is performed on dynamic clustering problem within the most widely used benchmark image and data sets by comparing it with the evolutionary algorithms, including DisABC, DCPSO and GA and the classical algorithms, including K-means and Fuzzy C-means. The obtained results clearly show that the proposed approach can satisfies both optimal number of clusters and well-obtained quality values simultaneously. It should be also notified that the CCP values are also presented to briefly show the quality of the algorithms in data clustering in addition to the values of VI index and obtained number of clusters. Moreover, the VI index which was first proposed for image clustering as an objective function is first time to be used in data clustering, to our knowledge. It is planned to apply the IDisABC algorithm to the different problems and to put forward a comprehensive comparative study on binary models.

References

[1] S. Rana, S. Jasola, R. Kumar, A review on particle swarm optimization algorithms and their applications to data clustering, Artif. Intell. Rev. 35 (2011) 211–222.
[2] A.K. Jain, M.N. Murty, P.J. Flynn, Data clustering: a review, ACM Comput. Surv. 31 (3) (1999) 264–323.
[3] R.H. Turi, Clustering-Based Colour Image Segmentation, Monash University, Australia, 2001.
[4] M.G.H. Omran, A. Salman, A.P. Engelbrecht, Dynamic clustering using particle swarm optimization with application in image segmentation, Pattern Anal. Appl. 8 (2006) 332–344.

[5] G. Hamerly, C. Elkan, Alternatives to the K-means algorithm that find better clusterings, in: Proceedings of the ACM Conference on Information and Knowledge Management (CIKM-2002), 2002, pp. 600–607.
[6] M. Omran, Particle Swarm Optimization Methods for Pattern Recognition and Image Processing, University of Pretoria, Environment and Information Technology, 2004.
[7] J. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proc. 5th Berkeley Symp. Math. Stat. Probability, 1967, pp. 281–297.
[8] J.C. Dunn, A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters, J. Cybern. 3 (1973) 32–57.
[9] R. Krishnapuram, J.M. Keller, A possibilistic approach to clustering, IEEE Trans. Fuzzy Syst. 1 (1993) 98–110.
[10] D.L. Pham, Spatial models for fuzzy clustering, Comput. Vis. Image Underst. 84 (2001) 285–297.
[11] Z. Deng, K.-S. Choi, F.-L. Chung, S. Wang, Enhanced soft subspace clustering integrating within-cluster and between-cluster information, Pattern Recognit. 43 (2010) 767–781.
[12] G. Gan, J. Wu, A convergence theorem for the fuzzy subspace clustering (FSC) algorithm, Pattern Recognit. 41 (2008) 1939–1947.
[13] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of the IEEE International Joint Conference on Neural Networks, Australia, 1995, pp. 1942–1948.
[14] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: 6th International Symposium on Micro Machine and Human Science, 1995.
[15] M. Omran, A. Engelbrecht, A. Salman, Particle swarm optimization method for image clustering, Int. J. Pattern Recognit. Artif. Intell. 19 (3) (2005) 297–322.
[16] M. Omran, S. Al-Sharhan, Barebones particle swarm methods for unsupervised image classification, in: IEEE Congress on Evolutionary Computation (CEC), 2007, 2007, pp. 3247–3252.
[17] W. Man To, H. Xiangjian, Y. Wei-Chang, Image clustering using particle swarm optimization, in: IEEE Congress on Evolutionary Computation (CEC), 2011, 2011, pp. 262–268.
[18] E. Hancer, C. Ozturk, D. Karaboga, Extraction of brain tumors from MRI images with artificial bee colony based segmentation methodology, in: 8th International Conference on Electrical and Electronics Engineering (ELECO), 2013, 2013, pp. 516–520.
[19] E. Hancer, C. Ozturk, D. Karaboga, Artificial Bee Colony based image clustering, in: IEEE Congress on Evolutionary Computation (CEC), 2012, Brisbane, Australia, 2012.
[20] C. Ozturk, E. Hancer, D. Karaboga, Improved clustering criterion for image clustering with artificial bee colony algorithm, Pattern Anal. Appl. (2014), http://dx.doi.org/10.1007/s10044-014-0365-y (in press).
[21] O.A.M. Jafar, R. Sivakumar, Ant-based clustering algorithms: a brief survey, Int. J. Comput. Theory Eng. 2 (2010) 787–796.
[22] T. Piatrik, E. Izquierdo, An application of ant colony optimization to image clustering, in: Proc. K-Space Jamboree Workshop, 2008.
[23] M. Abul Hasan, S. Ramakrishnan, A survey: hybrid evolutionary algorithms for cluster analysis, Artif. Intell. Rev. 36 (2011) 179–204.
[24] P. Berkhin, A survey of clustering data mining techniques, in: J. Kogan, C. Nicholas, M. Teboulle (Eds.), Grouping Multidimensional Data, Springer, Berlin, Heidelberg, 2006, pp. 25–71.
[25] E.R. Hruschka, R.J.G.B. Campello, A.A. Freitas, A.C.P.L.F. De Carvalho, A survey of evolutionary algorithms for clustering, IEEE Trans. Syst. Man Cybern. C: Appl. Rev. 39 (2009) 133–155.
[26] V. Kothari, J. Anuradha, S. Shah, P. Mittal, A survey on particle swarm optimization in feature selection, in: P.V. Krishna, M.R. Babu, E. Ariwa (Eds.), Global Trends in Information Systems and Software Applications, Springer, Berlin, Heidelberg, 2012, pp. 192–201.
[27] G. Ball, D. Hall, A clustering technique for summarizing multivariate data, Behav. Sci. 12 (1967) 153–155.
[28] J. Tou, R. Gonzalez, Pattern Recognition Principles, Addison-Wesley, Massachusetts, USA, 1979.
[29] C. Wallace, D. Dowe, Intrinsic classification by MML – the snob program, in: Seventh Australian Joint Conference on Artificial Intelligence, UNE, Armidale, NSW, Australia, 1994, pp. 37–44.
[30] L.I. Kuncheva, J.C. Bezdek, Nearest prototype classification: clustering, genetic algorithms, or random search? IEEE Trans. Syst. Man Cybern. C: Appl. Rev. 28 (1998) 160–164.
[31] S. Das, A. Abraham, A. Konar, Automatic clustering using an improved differential evolution algorithm, IEEE Trans. Syst. Man Cybern. C: Syst. Hum. 38 (2008) 218–237.
[32] S. Das, A. Konar, Automatic image pixel clustering with an improved differential evolution, Appl. Soft Comput. 9 (2009) 226–236.
[33] R.J. Kuo, Y.J. Syu, Z.-Y. Chen, F.C. Tien, Integration of particle swarm optimization and genetic algorithm for dynamic clustering, Inf. Sci. 195 (2012) 124–140.
[34] U. Maulik, I. Saha, Automatic fuzzy clustering using modified differential evolution for image classification, IEEE Trans. Geosci. Remote Sens. 48 (2010) 3503–3510.
[35] X. Rui, X. Jie, D.C. Wunsch, A comparison study of validity indices on swarm-intelligence-based clustering, IEEE Trans. Syst. Man Cybern. B 42 (2012) 1243–1256.
[36] M.H. Kashan, N. Nahavandi, A.H. Kashan, DisABC: a new artificial bee colony algorithm for binary optimization, Appl. Soft Comput. 12 (2012) 342–352.
[37] D. Karaboga, An idea based on honey bee swarm for numerical optimization, in: Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.
[38] D. Karaboga, B. Basturk, On the performance of artificial bee colony (ABC) algorithm, Appl. Soft Comput. 8 (2008) 687–697.
[39] D. Karaboga, C. Ozturk, A novel clustering approach: artificial bee colony (ABC) algorithm, Appl. Soft Comput. 11 (1) (2011) 652–657.
[40] D. Karaboga, C. Ozturk, Neural networks training by artificial bee colony algorithm on pattern classification, Neural Netw. World 19 (2009) 279–292.
[41] C. Ozturk, D. Karaboga, B. Gorkemli, Probabilistic dynamic deployment of wireless sensor networks by artificial bee colony algorithm, Sensors 11 (6) (2011) 6056–6065.
[42] D. Karaboga, S. Okdem, C. Ozturk, Cluster based wireless sensor network routing using artificial bee colony algorithm, Wireless Netw. 18 (7) (2012) 847–860.
[43] M. Ma, J. Liang, M. Guo, Y. Fan, Y. Yin, SAR image segmentation based on artificial bee colony algorithm, Appl. Soft Comput. 11 (2011) 5205–5214.
[44] D. Karaboga, B. Akay, A survey: algorithms simulating bee swarm intelligence, Artif. Intell. Rev. 31 (2009) 61–85.
[45] D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, A comprehensive survey: artificial bee colony (ABC) algorithm and applications, Artif. Intell. Rev. 42 (2012) 21–57.
[46] G. Pampara, A.P. Engelbrecht, Binary artificial bee colony optimization, in: IEEE Symposium on Swarm Intelligence (SIS), 2011, 2011, pp. 1–8.
[47] M.S. Kiran, M. Gunduz, XOR-based artificial bee colony algorithm for binary optimization, Turk. J. Electr. Eng. Comput. Sci. 21 (2013) 2307–2328.
[48] L. Wei, N. Ben, C. Hanning, Binary artificial bee colony algorithm for solving 0-1 knapsack problem, Adv. Inf. Sci. Serv. Sci. 4 (2012) 464–470.
[49] L. Wei, C. Hanning, BABC: a binary version of artificial bee colony algorithm for discrete optimization, Int. J. Adv. Comput. Technol. 4 (2012) 307–314.
[50] J.C. Dunn, Well separated clusters and optimal fuzzy partitions, J. Cybern. 4 (1974) 95–104.
[51] R.B. Calinski, J. Harabasz, A dendrite method for cluster analysis, Commun. Stat. 3 (1) (1974) 1–27.
[52] D.L. Davies, D.W. Bouldin, A cluster separation measure, IEEE Trans. Pattern Anal. Mach. Intell. 1 (1) (1979) 224–227.
[53] C.H. Chou, M.C. Su, E. Lai, A new cluster validity measure and its application to image compression, Pattern Anal. Appl. 7 (2) (2004) 205–220.
[54] M. Halkidi, Y. Batistakis, M. Vazirgiannis, On clustering validation techniques, J. Intell. Inf. Syst. 17 (2001) 107–145.