Combination of a New Metaheuristic Algorithm Based on Cooperative Grouper Fish - Octopus and DBSCAN Algorithm to Automatic Clustering

Alireza Balavand (a.balavand@srbiau.ac.ir)
Islamic Azad University https://orcid.org/0000-0002-6247-7184

Research Article

Keywords: Automatic Clustering, DBSCAN, Grouper Fish - Octopus Optimization Algorithm, Cluster Validity Indices

Posted Date: February 21st, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1118095/v1

Combination of a new metaheuristic algorithm based on cooperative grouper fish - octopus and DBSCAN algorithm to automatic clustering

Alireza Balavand [1]

Abstract

Density-based spatial clustering of applications with noise (DBSCAN) has been used to cluster data with arbitrary shapes which clustering is done based on the density among objects in data. Given that DBSCAN is a proper tool for identifying outliers and clustering non-convex data, it can be used for automatic clustering of non-convex data and covered the weakness of most automatic clustering algorithms in not recognizing non-convex clusters. So, in this paper, a new automatic clustering algorithm is introduced which is a combination of DBSCAN and grouper fish - octopus (GFO) algorithm. GFO-DBSCAN finds the best number of clusters in two main steps in an iterative manner. In the first step, the values of esp and $minpts$ are generated by GFO algorithm and in the second step, the clustering of data is performed using DBSCAN algorithm with eps and $minpts$ that are generated in the previous step. After each clustering, using correct data labels, and cluster centroids, the Calinski-Harabasz (CH) index is calculated. Finally, after passing some iterations of GFO, the best number of clusters is reported. In this study, three categories of data are used to measure the performance of the GFO-DBSCAN algorithm. Also, DBSCAN is compared with ACDE, DCPSO, and GCUK algorithms. According to the results, GFO-DBSCAN has achieved the optimal number of clusters in most data and has outperformed other well-known algorithms.

Keywords: Automatic Clustering, DBSCAN, Grouper Fish - Octopus Optimization Algorithm, Cluster Validity Indices

1. Introduction

Clustering is playing an important role in data mining, artificial intelligence, and other disciplines that are related to data science techniques. Using of clustering methods has been increasing day by day due to high speed, proper performance, and simplicity in implementation [1]. As an unsupervised technique, clustering places data in different clusters without any prior knowledge, that data with similar features are placed into each cluster [2]. Clustering methods have been divided into different categories that the most practical of them is included partitioning clustering, hierarchical clustering, and density-based clustering. In partitioning clustering, data is partitioned into K different groups with a predetermined number of clusters that clusters do not have any overlapping. K-means clustering algorithm [3] is the most popular in partitioning clustering that is easy to implement and has a high speed. In hierarchical clustering, a dendrogram (or tree structure) is generated and current clustering is created based on the previous step. Therefore, hierarchical clustering does not require any prior knowledge about the number of clusters [3]. Unlike partitioning and hierarchical methods that are based on the distance between objects in clusters and recognize convex clusters in most cases, density-based clustering is based on the density of data that can cluster data with arbitrary shapes. Density-based spatial clustering of applications with noise (DBSCAN) is the most practical density-based clustering algorithm that was proposed by [4]. DBSCAN can recognize the noise and outlier in data, which causes clustering is performed on arbitrary shapes with non-convex clusters.

Determining the optimum number of the cluster or automatic clustering has always been a challenge especially in arbitrary data. In recent years different algorithms have been proposed for automatic clustering [5]. The combination of partitioning clustering methods and metaheuristics has been had promising results compared to

* Corresponding author, E-mail: a.balavand@srbiau.ac.ir

Department of Industrial Engineering, Science and Research branch, Islamic Azad University
Hesarak Ave, Ashrafi Esfehani blvd, Tehran, Iran.

other methods [5]. Failure to identify the optimal number of clusters has been the main weakness of these methods in arbitrary data with non-convex clusters. So, in order to cover this weakness, GFO-DBSCAN method is proposed in this paper. This algorithm is a combination of grouper fish - octopus algorithm and DBSCAN in which automatic clustering is done with a different approach. In GFO-DBSCAN, CH index is considered as the objective function of GFO algorithm and DBSCAN is used for clustering. GFO-DBSCAN includes two main sections. In the first section, by using the GFO, the values of two parameters of eps and $minpts$ are generated and in the second section, clustering is done by DBSCAN and then the quality of clustering is investigated by CH validity index. The higher value of CH shows better cohesion and separation in clustering. Therefore, in the first phase, GFO algorithm generates decision variables, and in the second phase by using decision variables, clustering is done and the value of CH index is calculated. There are two factors that distinguish this article from other articles. Firstly, the combination of a density-based clustering algorithm and a metaheuristic algorithm is first used for automatic clustering. Secondly, noises and outliers are removed in the automatic clustering process by DBSCAN algorithm that has a big effect on distinguishing the optimum number of clusters in non-convex data.

The rest of this paper is organized as follows. In section 2, the literature review is investigated. The proposed algorithm is described in Section 3. In Section 4, the comparative results are presented and the conclusion is presented in section 5.

2. literature review

The automatic clustering methods based on metaheuristics are divided into single-solution-based algorithms, population-based algorithms, multiobjective algorithms. Population-based algorithms especially the kind of swarm intelligence have had better performance based on [5]. particle swarm optimization (PSO), ant colony optimization (ACO), invasive weed optimization (IWO), and bee colony optimization (BCO) have been often used for automatic clustering.

Metaheuristic algorithms based on Swarm intelligence have been had promising results in automatic clustering. PSO is the most practical of them that different algorithms have been developed based on it. Omran et al. [6] developed DCPSO that was a dynamic clustering method for image segmentation. DCPSO starts with a large cluster and selects the best number of the cluster by binary PSO. Then cluster centers have been refined by k-means algorithm. DI, TI, and SDbw were used as the objective function. The results of DCPSO was shown that the proper number of clusters was selected in most cases. MEPSO was proposed by Das et al. [7] that was based on a particle swarm-based segmentation algorithm for automatically grouping the pixels of an image into different homogeneous regions. In MEPSO, each particle was coded both cluster centers and the activation threshold. The objective function was a modified XB index. MEPSO was compared with Fuzzy-VGA [8] and fuzzy c-means [9]. MEPSO was used for clustering in [10] called KMEPSO in which kernelized CS index [10] was used as the objective function. GCUK [11], DCPSO, and conventional PSO were compared with KMEPSO that KMEPSO was better than other algorithms in terms of the quality of clustering. PSOPS was developed by Qu et al. [12]. Sym-index was used as the objective function and each particle was considered as cluster centers. PSOPS was compared with conventional PSO that was outperformed by conventional PSO when clusters were symmetric. PSOPS was not properly performed on non-symmetric clusters. Ant colony optimization (ACO) is another famous swarm intelligence population-based algorithm that was proposed by Dorigo [13]. Some methods of automatic clustering have been developed based on ACO. Ant-clustering [14] was developed in which refine cluster centers was done by fuzzy c-means algorithm. The performance of Ant-clustering was investigated by three real-world data that included Iris, Wine, and Glass. The results showed that Ant-clustering was able to find the best solution in all cases. ATTA-C was first proposed by Handl et al. [15]. ATTA-C by implementing some modifications on original ACO improved separation and cohesion results in clustering. Neighborhood function (NF) was used as the objective function of ATTA-C. ATTA-C was compared with k-means, SOM, and hierarchical clustering. The better results in automatic clustering belonged to ATTA-C compared to other methods. Invasive weed optimization (IWO) was proposed by Mehrabian and Lucas [16] that simulates the behavior of weed colonization. IWO-clustering [17] was the first algorithm basis on IWO. The solutions in IWO-clustering were called weeds. Each weed was coded by cluster centers and a modified Sym-index was used as the objective function. The performance of IWO-clustering was investigated by four real-world test functions. Also, IWO-clustering was compared with VGAPS [18], GCUK, and HNGA [19] that was outperformed these algorithms. Clustering with bee colony optimization (BCO) was done in [20]. Automatic clustering by BCO that called AKC-BCO was proposed by Kuo et al. [21] in which a Gaussian kernel and CS index were used as the objective function. In

AKC-BCO, solutions were divided into two sections that included threshold section and cluster centroids. The values with higher than 0.5 made the cluster be activated. Balavand et al. [2] proposed a combination of five cluster validity indices (CVI) and DEA for automatic clustering. That way that AP method is used to calculate the efficiency of efficient DMUs, so that maximum efficiency represented the optimal number of the cluster.

Density-based spatial clustering of applications with noise (DBSCAN) [4] is the most practical tool in the density-based clustering category. DBSCAN has been usually used for clustering data with arbitrary shapes corrupted with noise. In DBSCAN, determining the number of clusters is not required that is the main advantage of this method [22]. Determining two parameters of eps and $minpts$ is one of the challenges in DBSCAN algorithm [23]. Eps is the clustering radius and $minpts$ is the minimum number of clustering objects. Determining optimum values of these parameters leads to get different results in clustering. DBSCAN-GM [24] was proposed that was a combination of Gaussian-Means and DBSCAN algorithms. Gaussian-Means was used for generating small clusters with cluster centers to estimate parameters of DBSCAN. In Grid-based DBSCAN [25], the grid partition technique was used to estimate suitable parameters. Metaheuristic algorithms have had promising results to improve DBSCAN compare to traditional methods. In BDE-DBSCAN [26], binary Differential Evolution (DE) and DBSCAN were combined to estimate proper clustering parameters. PACA-DBSCAN [27] was proposed to data preprocessing phase in which a modified ant clustering algorithm (ACA) and design a new partitioning algorithm based on 'point density' (PD) were used. EWOA-DBSCAN [28] was used the earthworm optimization algorithm (EWOA) to determine proper values of DBSCAN's parameters for clustering data cube. Q Zhu et al. [29] proposed K-DBSCAN clustering method that by modifying the original harmony search optimization algorithm improved performance of DBSCAN in clustering.

3. Automatic clustering by GFO-DBSCAN algorithm

Automatic clustering algorithms based on population-based algorithms especially swarm intelligence methods have been used widely [5] because of their proper performance. How to encoding solutions and using different CVIs has been made different automatic clustering algorithms with different performance. The types of encoding schemes include binary, integer, and real schemes which solutions are often coded based on these encoding schemes in automatic clustering with metaheuristics. Binary encoding has been used to activate or deactivate clusters; integer encoding has been used to labeling data or cluster labeling; In real encoding, a solution has been shown a cluster center. CVIs evaluate the goodness of clustering using cohesion and separation of clusters [30]. Similarity within the cluster is calculated by cohesion and dissimilarity between clusters is calculated by separation. CVIs are often used as an objective function when automatic clustering is done by metaheuristics. In contrast, automatic clustering is done by different approaches in GFO-DBSCAN. In this algorithm, solutions are encoded by two decision variables. These variables are eps and $minpts$ that are input variables of DBSCAN. Determining the optimal values of these variables have a big effect on proper clustering in DBSCAN. In fact, whatever these variables are closer to the optimal value, the quality of clustering is increased. Automatic clustering is performed according to the following steps in GFO-DBSCAN. In the first step, the solutions are randomly generated in GFO algorithm. Each solution includes eps and $minpts$ that are between a lower and upper bound. In the second step, solutions are evaluated by the objective function. That way that at first clustering is performed by DBSCAN with values of eps and $minpts$ then the quality of clustering is measured using Calinski-Harabasz index (CH) [31]. Whatever the value of CH is higher, it is shown that the quality of clustering is better. In fact, the number of clusters is determined by DBSCAN which is directly related to values of eps and $minpts$. Finally, after passing a certain number of iterations, the best solution is reported by GFO in which the best solution represents the optimum number of clusters and the maximum value of CH. In the following, the GFO-DBSCAN will be explained in detail.

3.1 Grouper Fish and Octopus optimization algorithm (GFO)

In this section, a new optimization algorithm called the cooperative Grouper Fish and Octopus optimization algorithm (GFO) is proposed based on swarm intelligence. The GFO optimization algorithm is placed in the population-based algorithm that can solve optimization problems using swarm intelligence in continuous spaces. Like with other algorithms in this field, the GFO algorithm with exchanging information and creation an information flow through the collaboration of the members of the population reaches the optimum solution. This algorithm is based on the cooperative of Grouper Fish and Octopus that simulates their strategy in prey hunting according to Fig.1. This cooperative has been reported in [32] in which non-random association between grouper

fish and octopus has been proven in hunting the prey. This is a two-way partnership in which each hunter seeks to catch fish. Groupers and reef Octopuses are important predators on Indo-Pacific coral reefs and known as solitary hunters [32]. This collaboration is usually done as follows: the Groupers usually chase the fishes to hunt them. When fish hides in the coral reefs, Groupers can not be able to catch the fishes. So they signal octopuses using blue lights on their skin. When Octopuses receive the signals, they move towards the hunting location. Octopuses try to get out fishes from coral reefs and catch them and Groupers wait behind the octopuses and try to hunt fishes that octopuses can not catch them. This cooperation usually leads to success in hunting. Therefore, two phases of chasing and attacking are defined in the GFO algorithm. In the chasing phase, the Groupers chase the prey and usually direct the prey to the attacking area, and in the attacking phase, the Octopuses attack the prey. Also, a new crossover operation will be proposed that increases significantly the performance of GFO. In the following, two phases will be explained.

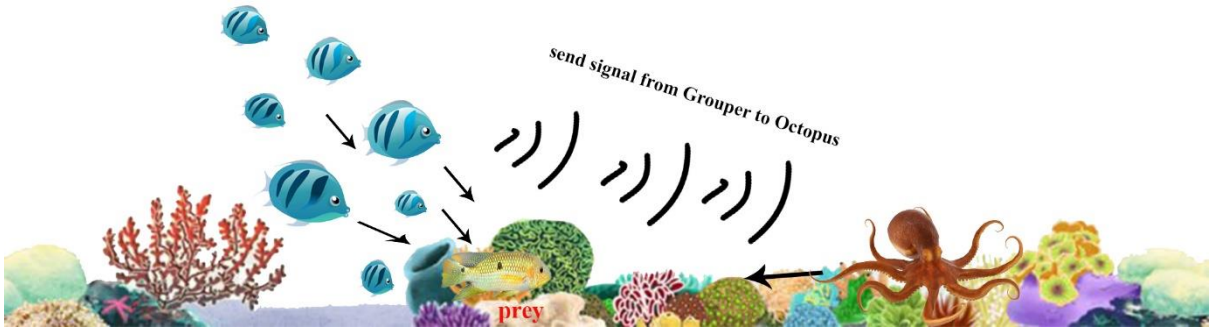

Fig.1. cooperation of groupers and octopus in hunting a prey

3.1.1 initializing

Like other metaheuristic algorithms before entering the main phases, the initializing phase is done in GFO. In the initializing phase, several initial solutions are randomly generated. In fact, these randomized solutions constitute the initial population of the GFO. These solutions are generated with uniform random distribution between the lower and upper bound. These solutions are generated based on Eq. (1).

$$x = LB + rand \times (UB - LB) \tag{1}$$

After determining initial parameters such as the number of population, the number of max iteration, the lower bound of variables, and the upper bounds of variables, initial population (x) are generated based on Eq. (1) where LB and UB are lower and upper bound respectively. Also, $rand$ is a uniform random variable that is generated between zero and one. Then these solutions are evaluated based on the objective function. In fact, in all phases of GFO, the solutions are evaluated based on the objective function. Then, the solutions are ascendingly sorted based on the objective function and the best solution is selected between all generated solutions. The best solution has the minimum objective function value that is known as the best solution (x_{best}).

3.1.2 chasing the prey

In this section, the behavior of the chasing of prey is simulated. Among the generated solutions, a solution is randomly selected as an octopus, and the rest of the solutions are known as groupers. According to the behavior of groupers, they follow the fish. In order to simulate the behavior of the groupers, Eq. (2) is proposed.

$$x_c^{i,t+1} = x_c^{i,t} + \beta\left(x_{best} - x_c^{i,t}\right) \ \forall i \tag{2}$$

In this equation, $x_c^{i,t}$ denotes to chaser i at iteration t where $i = (1,2,\dots,n_c)$ and $t = (1,2,\dots,maxiteration)$. x_{best} denotes the global best position that is known as the prey position. In this equation, β represents a random number that varies between 0 and 3. Eq. (2) shows that each chaser with a positive coefficient moves towards the prey position.

3.1.3 attacking the prey

The groupers chase fishes until catching them. They direct the prey to the coral reefs that the octopus is waiting for catching prey. Two states occur in catching the prey by groupers; in the first state, It is possible that groupers can not catch the prey and fishes hide in coral reefs, In that case, They send a signal to octopuses and octopuses move toward the prey. In the second step, groupers do not send any signals, and a random position is allocated to octopuses. In order to simulate this behavior, Eq.(3) is proposed. Eq (3) is divided into two sections. In the first section, if the Awareness signal (AS) is greater than or equal 0.5, each octopus attacks prey or x_{best}, otherwise, another section of Eq (3) is implemented in which a random position is generated for each octopus.

$$\begin{cases} x_d^{i,t+1} = x_d^{i,t} + r.\left(x_{best} - x_d^{i,t}\right) & \forall i\ AS \geq 0.5 \\ x_d^{i,t+1} = random\ position\ \forall i & AS < \alpha \end{cases} \quad (3)$$

Based on Eq. (3), $x_d^{i,t+1}$ denotes to octopus i at iteration $t+1$ where $i = (1,2,\dots,n_d)$. $t+1$ denotes the new solution that generates based on Eq. (3) at iteration t. x_{best} shows the best solution and r shows a random number that generates based on the uniform distribution. In this Equation, For generating a new solution in the first section of Eq. (3), each octopus with a positive random coefficient (r) moves toward the prey. In the second section of Eq. (3), a random position is generated for each octopus according to the pseudo-code of Fig. 1.

For generating random positions, the following pseudo-code is implemented.

1. $n = generate\ random\ integer\ number$ between 1 and max number of variables
2. $n1 = randperm(\max number\ of\ variables, \mathrm{n})$
3. $n2 = randperm(\max number\ of\ variables, \mathrm{n})$
4. **For** k=1:number element of n1
5. $\quad Pop.position(n1(k)) = Best.position(n2(k))$
6. **End for**

Fig.2. pseudo-code of crossover operation

According to Fig.2, a random integer number is generated between 1 and max number of decision variables in line1. In lines 2,3, To the number of n, two permutation vectors are generated as $(n1\ and\ n2)$. In lines 4 to 6, each element of the octopus position is replaced with a different element of the best position. This crossover operation increases significantly the performance of GFO that is proposed in this paper for the first time.

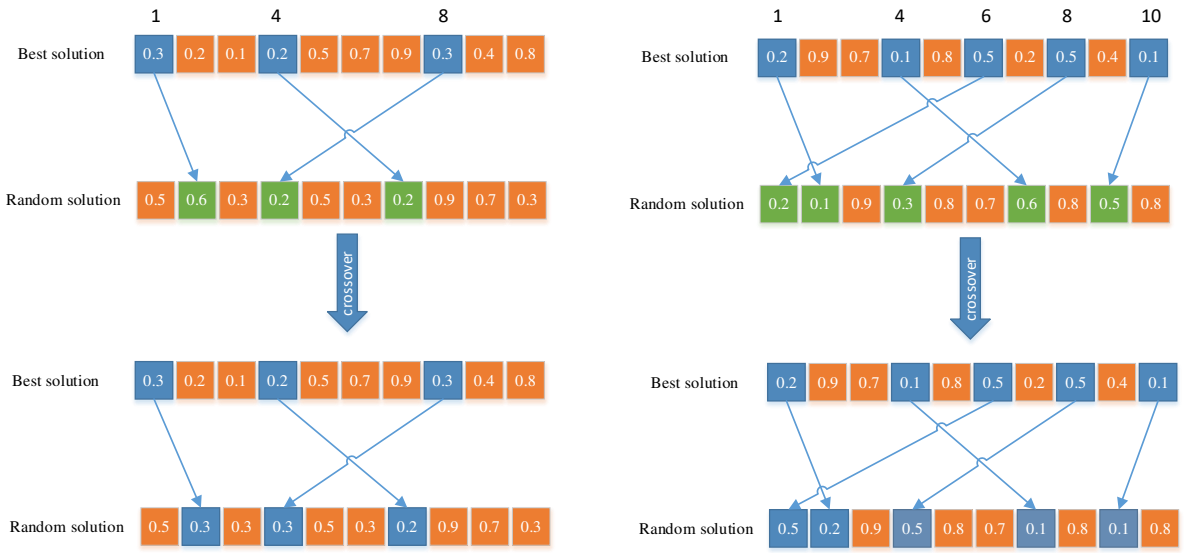

Fig.3. crossover operation process

The crossover process is shown in Fig.3 with two random solutions. According to the top left section of this figure, the crossover is done between the best solution and a selected random solution. A permutation vector with three members is generated. This vector includes 1, 4, and 8 that the indexes corresponding to this vector are 0.3, 0.2, and 0.3. These selected elements are replaced instead of first, fourth, and eighth members of the random solution according to the bottom left section of Fig.3. According to the top right section of Fig.3, a new permutation vector is generated that the members of this vector are 1, 4, 6, 8, and 10. Indexes corresponding to this vector are 0.2, 0.1, 0.5, 0.5 and 0.1. These selected members of the best solution are replaced instead of the random solution according to the right bottom section of Fig.3. This new method is proposed for the first time in this paper that has a significant effect on the performance of GFO.

3.2 pseudocode of the GFO

Given that the GFO algorithm is implemented in the Matlab software 2020b, in order to more explain about structure of the programming of the GFO algorithm, the pseudocode is presented in Fig. 4. According to the pseudocode, after determining the initial parameters, such as the number of population, the maximum number of the iteration, the number of population of each group, the lower and upper bound of the decision variable, and etc., the initializing phase is done. In the initializing phase, to the number of the population, the random solutions are generated based on the uniform distribution that all of the solutions are between lower and upper bound. These random solutions are evaluated based on the objective function. Before entering the main phases, the population is divided into two groups. Among the population, the solution with the minimum value of the objective function is considered as the best solution or the prey. Then among the rest of the population, 50 percent of the population is considered as the chasers or grouper fishes, and 50 percent of the rest of the population is considered as the attackers or octopuses. It is assumed that the solution with the minimum objective value is the best solution and other members of the population are considered as the hunters. Therefore, the population number is equal to $npop = nc + ng$ which is always an odd number. Because the number of the population includes the best solution (ng) and the hunters' groups (nc) and given that the population of hunters is always an even number, the number of all members of the population is always an odd number. In the following of the pseudocode, the first phase is performed based on Eq. (2) for each solution of the chasers population. Then each solution is evaluated based on the objective function. In this phase, if a better solution is found, it will be replaced instead of the best solution. In the second phase, for each solution of the attackers' population, Eq. (3) is performed and each solution is evaluated. In this phase, a better solution will be replaced instead of the best solution. After passing all phases of the algorithm, the stopping criteria are checked and if the stopping criteria are satisfied, the best solutions, along with the value of the objective value will be reported. Otherwise, the number of iterations is added to a unit, and from the first phase, all steps are repeated.

```
set initial parameters
best solution= +∞
 • Initializing
  For each population
   Implementing of Eq.(1)
   Generate random position between lower bound and upper  bound
   Evaluation of random position
   Update the best solution and current solution
  end if
 End for
 update the best solution
 • Main loop
 dividing the population into the chasers, and the attacker
For i=1 to it
-First phase
  For each population of NC
    Implementing of Eq.(2)
      Update the best solution and current solution
     end if
 End for
-Second phase
  For each population of ND
    If AC<=0.5
      Implementing of the top section of Eq.(3)
   Esleif AC>0.5
      Generate random position according to pseudocode of Fig.2
   end if
 End for
It=it+1
End for (main loop)
Report the best solution
```

Fig. 4. the pseudocode of the FOA

3.3 exploration and exploitation

In designing a metaheuristic algorithm, the balancing of the two criteria of exploration and exploitation [33] [34] [35], increase the efficiency of the algorithm. The experiences show that in the initial iterations, the ability of exploration should be increased and in the final iterations, the power of exploitation should be increased. This means that in the initial iterations, the different parts of the space solution should be searched, and in the last iterations, around the optimal solution should be searched more accurately. This rule has been observed in the GFO algorithm. exploration is performed in the Eq. (2) related to the first phase and in the above section of Eq. (3) related to the second phase. In bottom section of Eq. (3), exploitation is done in which a new crossover method has been proposed. In this way, in the exploitation phases, small mutations occur and in the exploration phases, longer mutations occur.

3.4 Clustering by DBSCAN

DBSCAN has been used to cluster non-convex data with noises. DBSCAN is not based on the distance between objects and density among data is considered in DBSCAN. Determining the optimal values of eps and $minpts$ lead to better clustering in DBSCAN. eps is the radius of the adjacent neighborhood and $minpts$ is the minimum number of data points that are located in the adjacent neighborhood. So, it is possible that with different values of

eps and $minpts$, The number of different clusters is obtained. The clustering process of DBSCAN algorithm is according to Fig. 5.

```
set initial parameters D, eps, minpts
init C=0
For each unvisited point p in D
  Mark p as visited point
  N=GetNeighbours(p,eps)
   If size(N)<minpts
     Mark p as noise
   Elseif size(N)>=minpts
     C=nextcluster
     ExpandCluster (p, N, C, Eps, MinPts)
  Endif
Endfor
Report cluster labels and outliers data
```

```
ExpandCluster (p, N, C, Eps, MinPts)
  Add p to cluster C
 For each point q in N
  If q is not visited
    Mark q as visited
    N'= GetNeighbours(q,eps)
    If size(N')>=minpts
      N=N joined with N'
   Endif
  If q is not yet a member of any cluster
      Add q to cluster C
 Endif
Endfor
```

Fig. 5. Clustering process by DBSCAN

According to pseudo-code, initialization is done in which D (Dataset), eps and $minpts$ are initialized. Then C (cluster) is considered zero. In a for loop, each point (p) in D is visited and then the neighbors of each point are calculated that these neighbors are in the N. The size of N is checked and if N is smaller than $minpts$, p is knowns as the outlier. If N is bigger than $minpts$, Cluster C is created and *ExpandCluster* function is implemented. The output of DBSCAN is labels that show that each data belongs to a different cluster or are outliers.

It is difficult to predict the values of eps and $minpts$. So, metaheuristics are used as a remedy to determine optimum values of eps and $minpts$. In many papers that clustering has been done by the combination of DBSCAN and metaheuristics, Two important points have been considered; the values of eps and $minpts$ should be selected in such a way that most of the data detect as valid clusters [36] and the number of clusters is not exceeded from the predetermined number of clusters that are determined by the user [29]. So, by iterative manner, the optimum values of eps and $minpts$ can be obtained by metaheuristics.

3.5 Calinski-Harabasz index (CH)

This index has been proposed for crisp clustering that mathematical equation is according to Eq . (4):

$$CH(C) = \frac{N-K}{K-1} \times \frac{\sum_{c_k \in C} n_k d_e(\bar{c}_k, \bar{X})}{\sum_{c_k \in C} \sum_{x_i \in c_k} d_e(\mathbf{x}_i, \bar{c}_k)} \tag{4}$$

where $\boldsymbol{x_i} \in \mathbb{R}$ and shows the object**,** N is numver of objext in data, $\boldsymbol{X} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_N}\}$ shows a dataset, $\bar{\boldsymbol{c}}_{\boldsymbol{k}}$ is cluster center that the mathematical equation is $\bar{\boldsymbol{c}}_{\boldsymbol{k}} = \frac{1}{\boldsymbol{n_k} \sum_{\boldsymbol{x_i} \in \boldsymbol{c_k}} \boldsymbol{x_i}}$**,** dataset center is defined as $\bar{\boldsymbol{X}} = \frac{1}{\boldsymbol{N} \sum_{\boldsymbol{x_i} \in \boldsymbol{X}} \boldsymbol{x_i}}$, $\boldsymbol{n_k}$ is the number of objects in the cluster $\boldsymbol{c_k}$, and K is the number of clusters. At the numerator of CH, the sum of the distance between cluster center and dataset center is calculated and in the denominator of the CH, sum of the distance between cluster center and points is calculated. In fact, cohesion is calculated in the denominator, and separation is calculated in the numerator.

3.6 Combination of GFO and DBSCAN

In this section, the combination of GFO and DBSCAN will be explained. As mentioned before, GFO-DBSCAN has two phases according to Fig. 6. In the left section of Fig.6, the variables of eps and $minpts$ are generated by GFO algorithm and in the right section of Fig. 6, the variables are evaluated with the cost function. GFO algorithm is divided into two main phases of the initialization phase and the main loop phase. In the initialization phase, the values of eps and $minpts$ are generated between the lower and upper bound according to line 4 of GFO section, and these solutions are evaluated by the cost function. In the cost function, there are three inputs of D (data), eps, and $minpts$. According to line 4 of the cost function, clustering is done by DBSCAN algorithm and the labels are generated by DBSCAN that called IDX. The percent of noises and the number of clusters are calculated in the cost function in lines 5 and 6 respectively. In order to prevent creating small clusters and a high number of noises,

lines 7 to 9 of the cost function are implemented. By implementing lines 7-9, a penalty value is added to the cost function which makes that solutions with a high percent of noises or clusters smaller than 0.1 of number of all data are not considered. In line 10 of the cost function, the cluster centroids, and in line 11, the distance between objects of D are calculated. In line 12 of the cost function, CH index is calculated and multiplied by a negative number. Because the minimization is done in GFO algorithm. In line 13, the solutions are reported. In the main loop of GFO algorithm, the values of eps and $minpts$ are generated and evaluated by the cost function. In all sections of GFO, as soon as a better solution is found, it is replaced instead of the best solution. finally, the global best solution with a minimum CH index is reported as the best solution.

In order to measure distance among objects, several methods have been proposed in the clustering problem, selection of the distance method is very important because it can create different clusters in clustering [37]. the Minkowski metric [38] has been widely used in many papers. The Minkowski metric with $p = 2$ is used in this paper that its equation is according Eq. (5):

$$d_p(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{D} |x_i - y_i|^p \right)^{\frac{1}{p}} \tag{5}$$

In Eq (5), city block [39] with $p = 1$, the, euclidean metric with $p = 2$, and Chebyshev metric [40] with $\boldsymbol{p} = \infty$, are calculated.

GFO algorithm	***Cost function***
1 set initial parameters	*1 Input parameters D, eps, minpts*
2 best solution=+∞	*2 C=0*
• ***Initializing***	*3 Penalty=0*
3 ***For*** *each population*	*4 IDX=dbscan(D,eps,minpts)*
4 Generate random values of eps and minpts	*5 Percent_of_noises=*$\frac{n\_noises}{n\_data}$
5 Evaluation clustering by ***Cost function***	*6 N_clusters=Calculate number of clusters*
6 Update the best solution and current solution	*7* ***If*** *or (*$\frac{number\ of\ object\ of\ each\ cluster}{n\_data}$ *<0.1, Percent_of_noises)*
7 ***End for***	*8 C=C+penalty*
8 dividing the population into the chasers and ambushers	*9 End*
• ***Main loop***	*10 M=mean of each cluster*
9 ***While it*** *≤* ***maxit***	*11 Dis=calculate distance between objects of D*
First phase	*12 CH=validityindices(Dis,M)*
10 ***For*** *each population of chasers*	*13 Report -CH, number of clusters, eps, minpts, and IDX*
11 Generate eps and minpts Accrding to Eq.(2)	
12 evaluate solutions by ***Cost function***	
13 Update the best solution and current solution	
14 ***End for***	
Second phase	
15 ***For*** *each population of attackers*	
16 ***If*** $(AC) \leq 0.5$	
17 Implementing the above section of Eq.(3)	
18 ***Else if*** $(AC) > 0.5$	
19 Implementing bottom section of Eq.(3)	
20 ***end if***	
21 evaluate solutions by ***Cost function***	
22 Update the best solution and current solution	
23 ***End for***	
24 It=it+1	
25 ***End while (main loop)***	
26 Report the best solution	

Fig. 6. Combination of GFO algorithm and DBSCAN clustering

4. Investigation of GFO-DBSCAN

This section is divided into two sections. In the first section, GFO is investigated by classical benchmark functions and in the second section, GFO-DBSCAN is evaluated by well-known data sets. Standard data sets are the best tool to examine the performance of automatic clustering algorithms. Data sets are divided into two categories: synthetic data and real-life data. In most papers, synthetic data sets are preferred because of linear separability between clusters and the complexity of the data. Three categories of synthetic data sets are defined: linear separability without overlapping, linear separability with overlapping, non-linear separability without

overlapping. Given that the purpose of implementing GFO-DBSCAN algorithm, identify the optimal number of clusters in arbitrary data, synthetic data is used for comparing the performance of GFO-DBSCAN. In this section, GFO-DBSCAN is evaluated with three data sets. These data sets are summarized according to Table 1. In this table, the kind of data, the number of points, and the optimal cluster number are shown. According to this table, synthetic data sets are the first category, the second category includes convex data sets and three data including Iris, Wine, Breast-Cancer, and Vowel are categorized as real data sets. In this section, the performance of GFO-DBSCAN will be investigated with three kinds of data sets and compared with four well-known automatic clustering algorithms.

4.1 Evaluation of the GFO algorithm

In this section, eight well-known classical test functions are used to evaluate the efficiency of the GFO algorithm. In the following, classical test functions will be explained and metaheuristics will be compared using test functions. In Table 1, eight well-known test functions are shown that these functions are used to evaluate the GFO algorithm. In Table 1, All benchmark functions are investigated in thirty dimensions. In this table, the code of each test function, the name of each function, the lower and upper bound of each function, the dimensions investigated, the formula, the minimum value of each function, and the type of each function in terms of unimodal and multimodal are shown respectively. According to Table 1, there are five multimodal and three unimodal functions. In particular, unimodal functions are useful tools for benchmarking exploitation. In contrast, multimodal functions have many local optima and are suitable tools to evaluate the ability of exploration of metaheuristics.

Table 1. Thirty Dimension Benchmark functions (NF: Number of Functions, D: Dimension, U: Unimodal, M = Multimodal)

NF	Function	Range	D	Formulation	Min	type
(f_1)	Ackley	$[-35,35]$	30	$f(x) = -20\exp\left[-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right] - \exp\left[\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right] + 20 + \exp(1)$	0	M
(f_2)	Griewank	$[-600,600]$	30	$f(x) = \sum_{i=1}^{n}\frac{x_i^2}{4000} - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	0	M
(f_3)	Michalewicz	$[0,\pi]$	30	$f(x) = -\sum_{i=1}^{n} sin(x_i)\, sin^{2m}\left(\frac{ix_i^2}{\pi}\right)$	-29.630	M
(f_4)	Rastrigin	$[-5.12,5.12]$	30	$f(x) = 10n + \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i)\right]$	0	M
(f_5)	Rosenbrock	$[-100,100]$	30	$f(x) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	0	M
(f_6)	Sphere	$[-100,100]$	30	$f(x) = \sum_{i=1}^{n} x_i^2$	0	U
(f_7)	Step	$[-5.12,5.12]$	30	$f(x) = \sum_{i=1}^{n}(x_i + 0.5)^2$	0	U
(f_8)	Quartic	$[-1.28,1.28]$		$f(x) = \sum_{i=1}^{n} ix_i^4 + rand(0,1)$	0	U

Perspective views of the classic test functions are displayed in Fig.7. In this study, the number of function evaluations (NFE) is used for the stopping criteria. The value of NFE is equal to $NFE = maxit \times npop \times nloop$ that $maxit$ denotes a maximum of iteration, $npop$ denotes the number of population of the algorithm and $nloop$ denotes the number of loops of the algorithm. All algorithms have been evaluated with 180,000 number of NFE. In order to validate the results, the GFO algorithm is compared with four well-known high-performance algorithms that are Ant Colony Optimization (ACO) [41], Differential Evolution (DE) [42], Gray Wolf Optimization (GWO) [18], and Particle Swarm Optimization (PSO) [43]. Table 2 shows the results of the study on classical functions. In this table, three indicators of the best cost, the average of the best costs, the variance of the best costs have been used to investigate the performance of the GFO algorithm in the functions of f_1 to f_8. The average of the best cost is calculated based on $A = \frac{\sum_{i=1}^{maxit} bestcost_i}{maxit}$ that $bestcost_i$ is the value of the best cost

in ith iteration. Also, the variance of the best cost is calculated based on $var(bestcosts)$ which $bestcosts$ is the vector of the best costs. In this table, the functions of Ackley, Griewank, Michalewicz, Rastrigin, Rosenbrock, Sphere, Step, and Quartic are investigated in the thirty dimensions. The Ackley function has plenty of local minimal and a large hole is located in the middle of this function. The Griewank function has a lot of local minimal. The specific structure of the Michalewicz function makes that this function becomes a hard function. The Rastrigin function is a high multimodal function, but its local minima are distributed regularly. The well-known function of Rosenbrock is a hard function that converging to its optimal global solution has always been a challenge for metaheuristics. The sphere function is unimodal and convex. The Step function is the same as the Sphere function, with the difference that its center is moved. The Quartic function is a unimodal function that is always a hard function among convex functions.

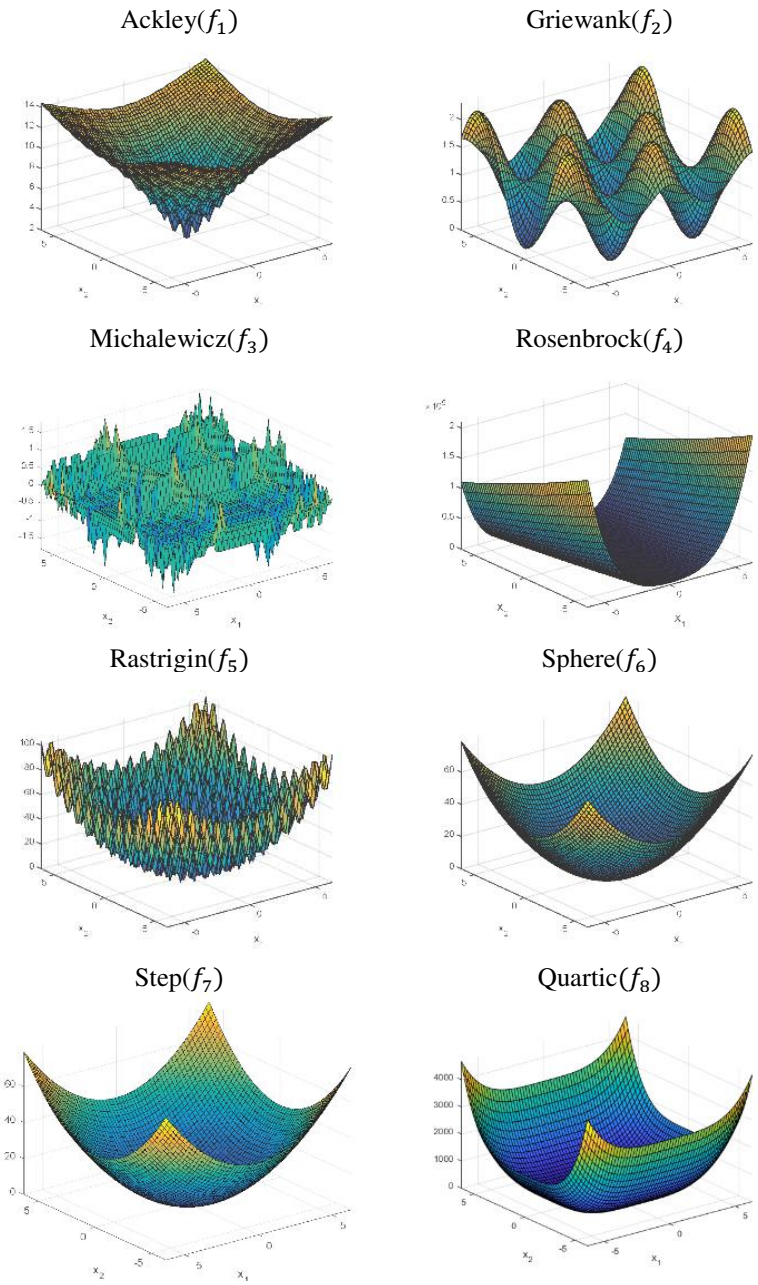

Fig. 7. Perspective view for benchmark functions

The search history of the GFO algorithm in Ackley, Michalewicz, Rastrigin, and Quartic is shown in Fig. 8. This figure shows that the GFO algorithm uses exploration to search almost all local optimizations and searches for better solutions after approaching the best position. The obtained results in Table 2 show that in all test functions, the GFO algorithm has had superior performance. Based on this table, GFO algorithm has reached better solutions

than other algorithms in the functions of f_2, f_4, f_5, f_6, f_7, and f_8 in terms of best solutions. Furthermore, the GFO algorithm has reached superior performance than other benchmark functions of $f_1, f_2, f_4, f_5, f_6, f_7$, and f_8 in terms of the average of best solutions. In functions of f_1, f_2, f_5, f_6, f_7, and f_8, GFO has had better performance compared to other benchmark functions. The extreme convergence of the GFO algorithm in benchmark functions is quite obvious evident in Fig. 9. In total, among the eight classic functions, the GFO algorithm has reached a better performance than other benchmark functions that these results indicate a good balance between the exploration and exploitation in the GFO algorithm.

Table 2. Comparative results of the GFO with ACO, DE, GA, and PSO in three-dimensional benchmark functions

Function	parameters	ACO	DE	GFO	GWO	PSO
f_1	Best	**4.4409E-15 (1)**	7.9936E-15 (2)	7.9936E-15 (2)	7.9936E-15b (2)	1.5099E-14 (3)
	Mean	2.3881E-01 (4)	6.8973E-01 (5)	**3.2445E-02 (1)**	9.2344E-02 (2)	1.9889E-01 (3)
	Variance	2.8412E+00 (4)	8.5088E+00 (5)	**2.3421E-01 (1)**	1.1951E+00 (3)	1.1346E+00 (2)
f_2	Best	**0.0000E+00 (1)**	**0.0000E+00 (1)**	**0.0000E+00 (1)**	**0.0000E+00 (1)**	1.4780E-02 (2)
	Mean	1.5130E+00 (4)	4.6463E+00 (5)	**7.9425E-02 (1)**	8.5570E-01 (3)	4.2347E-01 (2)
	Variance	3.8211E+02 (4)	1.3931E+03 (5)	**3.5460E+00 (1)**	2.9826E+02 (3)	5.8652E+01 (2)
f_3	Best	-1.5924E+01 (4)	-2.7363E+01 (2)	-2.6860E+01 (3)	-1.2900E+01 (5)	**-2.8482E+01 (1)**
	Mean	-1.4737E+01 (4)	-2.4206E+01 (3)	-2.5316E+01 (2)	-1.0065E+01 (5)	**-2.7933E+01 (1)**
	Variance	2.8995E+00 (2)	1.3713E+01 (5)	5.9737E+00 (4)	**1.4397E+00 (1)**	5.9456E+00 (3)
f_4	Best	2.9849E+00 (3)	1.8498E+01 (4)	**0.0000E+00 (1)**	0.0000E+00 (1)	3.6813E+01 (5)
	Mean	3.0409E+01 (3)	5.6308E+01 (5)	**1.7245E+00 (1)**	3.1025E+00 (2)	3.8645E+01 (4)
	Variance	4.2371E+03 (5)	2.2600E+03 (4)	2.6024E+02 (2)	6.7918E+02 (3)	**2.5945E+02 (1)**
f_5	Best	2.1943E+01 (3)	3.2256E+01 (5)	**5.8979E-24 (1)**	2.6151E+01 (4)	1.3903E+01 (2)
	Mean	6.3075E+07 (4)	1.3734E+08 (5)	**4.9235E+05 (1)**	2.2851E+07 (3)	4.4951E+06 (2)
	Variance	1.0694E+18 (4)	2.1297E+18 (5)	**3.7834E+14 (1)**	3.9069E+17 (3)	2.9402E+16 (2)
f_6	Best	1.6124E-130 (2)	8.3734E-54 (4)	**0.0000E+00 (1)**	0.0000E+00 (1)	1.4182E-79 (3)
	Mean	1.6012E+02 (4)	4.7869E+02 (5)	**8.4491E+00 (1)**	8.0299E+01 (3)	3.5657E+01 (2)
	Variance	4.7831E+06 (4)	1.5132E+07 (5)	**7.1506E+04 (1)**	2.8112E+06 (3)	6.1902E+05 (2)
f_7	Best	0.0000E+00 (1)	0.0000E+00 (1)	**0.0000E+00 (1)**	5.2850E-01 (2)	0.0000E+00 (1)
	Mean	4.9665E-01 (3)	1.2083E+00 (5)	**1.9051E-02 (1)**	1.1846E+00 (4)	9.3331E-02 (2)
	Variance	4.0133E+01 (4)	1.0157E+02 (5)	**3.5783E-01 (1)**	2.3854E+01 (3)	5.7536E+00 (2)
f_8	Best	1.6650E-03 (3)	5.8355E-03 (5)	**1.2632E-05 (1)**	5.9721E-05 (2)	1.9529E-03 (4)
	Mean	1.8810E-01 (4)	3.7271E-01 (5)	**2.7087E-03 (1)**	1.2896E-01 (3)	2.8760E-02 (2)
	Variance	7.2838E+00 (3)	1.2967E+01 (5)	**7.8103E-03 (1)**	1.1361E+01 (4)	5.8099E-01 (2)

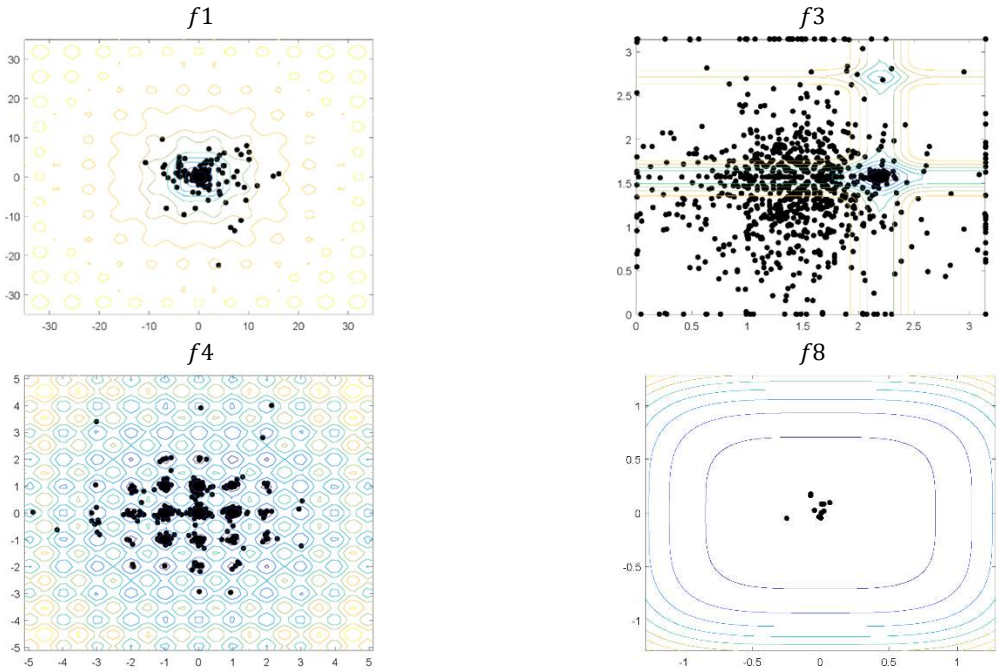

Fig. 8. Search history of the GFO

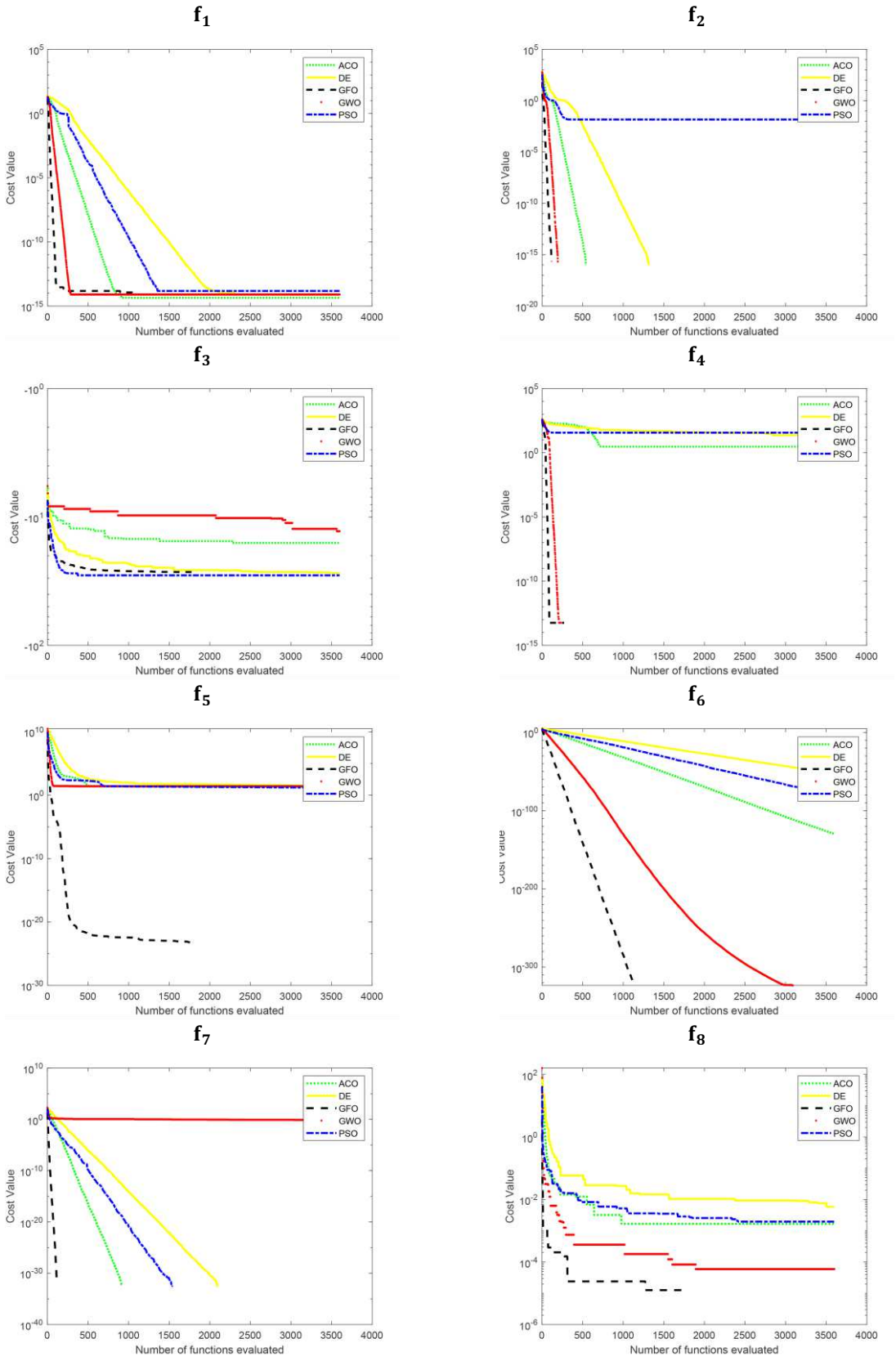

Fig. 9. Compare performance algorithms

4.2 Experimental results of GFO-DBSACN in synthetic data sets

Most synthetic data are non-convex and have been considered as proper benchmarks for the investigation performance of clustering algorithms. Whereas automatic clustering algorithms have been often investigated by convex data because most of them are based on the distance between objects and identifying the non-convex clusters has been a challenge for them and comparative results in non-convex have not been shown good results in research papers. So, sample datasets with convex shapes are often preferred in most cases. Given that GFO-DBSCAN is based on the density of data, six synthetic data sets are used for the investigation performance of GFO-DBSCAN in which most data sets have non-convex clusters. Linear separability without overlapping with non-convex clusters is the main feature in most of these data sets. These data sets include Aggregation [44], R15 [45], Jain [46], Spiral [47], Pathbased [47], and Flame [48]. Table 3 shows the three kinds of data sets that synthetic data sets are located in the first section of this table. According to Table 3, Aggregation data has the maximum number of records among data sets with 788 records. The best number of clusters in aggregation is seven and this data has a non-convex cluster. R15 has 600 records and the best number of clusters for this data is 15 and all of the clusters in R15 are convex. The number of records in Jain is 373 and two clusters are the best number of clusters for this data. The largest cluster of Jain is non-convex. Spiral has 312 records and three clusters are optimum for this data. Pathbased has 300 records and three optimum clusters. The largest cluster of this data is quite non-convex. Flame has the minimum number of data among data sets with two optimal clusters that one of the clusters of this data is non-convex. Table 4 shows the initial parameters setting of GFO-DBSCAN algorithm. $npop$ represents the number of population, $maxit$ denotes the maximum number of iterations, $nchaser$ shows the number of chasers' population, $nattacker$ denotes the number of attackers' population, $EpsLB$ shows the lower bound of eps, $EpsUB$ represents the upper bound of eps, $MinptsLB$ shows the lower bound of $minpts$ and $MinptsLB$ represents the upper bound of $minpts$. The values of SDR and LDR are the smallest and largest values of each data set according to [29]. num_obj represents the number of all objects, K shows the maximum number of clusters, and D denotes the number of dimensions of data.

Table 3. Test data sets

Group of data	Dataset	Number of data point	The optimal number of cluster
Synthetic data sets	Aggregation	788	7
	R15	600	15
	Jain	373	2
	Spiral	312	3
	Pathbased	300	3
	Flame	240	2
Convex data	Data-2	200	2
	Data-3	250	3
	Data-4	400	4
	Data-5	500	5
	S1	5000	15
	S2	5000	15
Real data	Iris	3	150
	Wine	3	178
	Breast-Cancer	2	699

Table 4. parameter setting for GFO algorithm

Parameters	Values
$npop$	30
$maxit$	20
$nchaser$	15
$nattacker$	15
$EpsLB$	0
$EpsUB$	$\frac{SDR}{2 \times K}$
$MinptsLB$	1
$MinptsUB$	$\frac{num\_obj}{\frac{LDR}{SDR} \times K \times D}$

Table 5 represents the performance of GFO-DBSCAN in six synthetic data sets and Fig 10 shows the results of automatic clustering of DB-SCAN schematically. According to Fig 10., each cluster is marked with different colors, and outliers are highlighted in red. The value of esp, $minpts$, the number of best clusters, and the cost of GFO algorithm are shown in this table. For instance, in Aggregation data, the value of eps and $minpts$ are 1.7532 and 11 respectively. Also, GFO-DBSCAN has identified seven clusters as the best number of clusters in Aggregation data that are the optimal number of clusters according to Table 3. In the last column of Table 3, the cost of GFO algorithm is shown that the cost value is the minimum value of CH that has reached during the different iteration of GFO. In fact, less value of cost shows the better clustering which GFO is achieved minimum value of CH in a certain number of clusters. In the optimum cost, the value of separation and cohesion are maximum and minimum respectively. According to Table 5, GFO-DBSCAN has achieved the best number of clusters in all data. These results show the superior performance of GFO-DBSCAN in non-convex data. The clustering by CHD-DBSCAN is shown in all synthetic data in Fig. 10. Proper performance of GFO-DBSCAN is quite shown especially in non-convex data. According to Fig. 4, GFO-DBSCAN has obtained the optimum solution in most data sets. In Jain and Spiral data, it is obvious that the optimum solution has not been obtained. However, the optimal number of clusters has been identified. The outliers are shown by the red points or -1 in the legend of Fig.4. Identifying outliers by DBSCAN algorithm has had a big effect in converging optimal solutions in automatic clustering. In Fig.11, the number of clusters is shown that GFO-DBSCAN has obtained in 20 iterations in synthetic data. For instance, in R15, two clusters have been obtained in the first iteration as the best number of clusters and 15 clusters are obtained from iteration two to 20 as the best number of clusters. In Aggregation, the best number of clusters has been obtained from the first iteration.

Table 5. the performance of GFO-DBSCAN in synthetic data sets.

Data	**The optimal value of eps**	**The optimal value of $minpts$**	**Number of the best cluster**	**Cost of GFO**
Aggregation	1.7532	11	7	-131.3452
R15	0.5	14	15	-1384.7432
Jain	2.7381	5	2	-63.4596
Spiral	3.4991	8	3	-8.4336
Pathbased	1.7974	4	3	-57.5898
Flame	1.3587	11	2	-107.7713

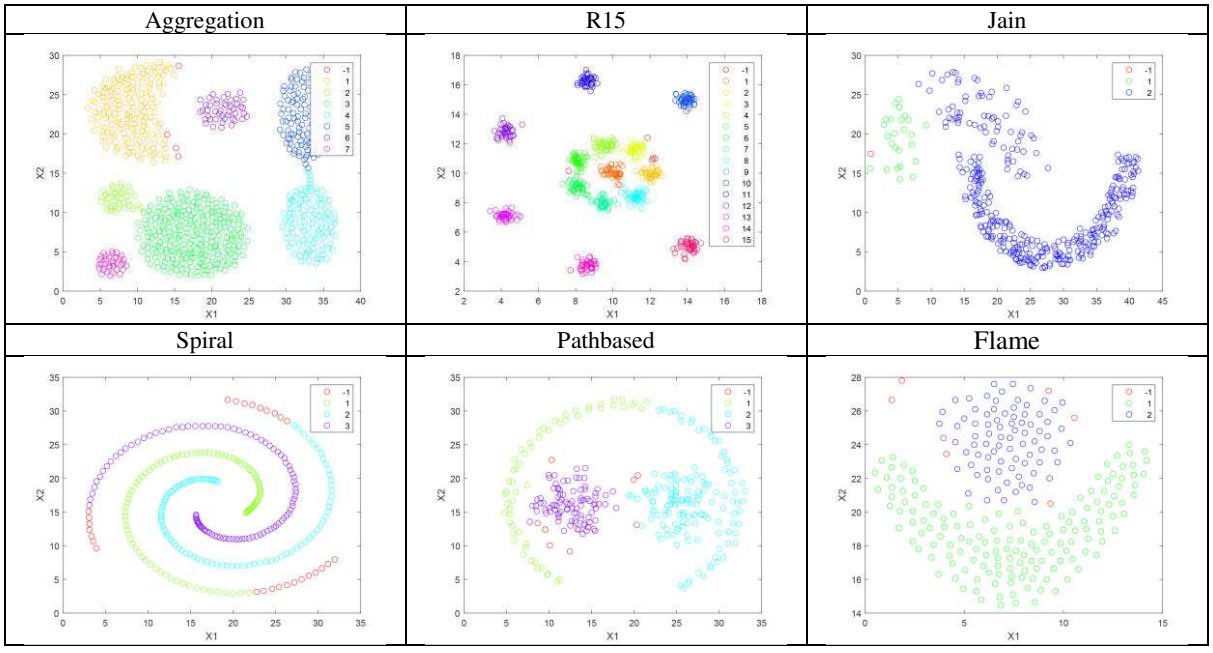

Fig. 10. Automatic clustering by GFO-DBSCAN in synthetic data sets.

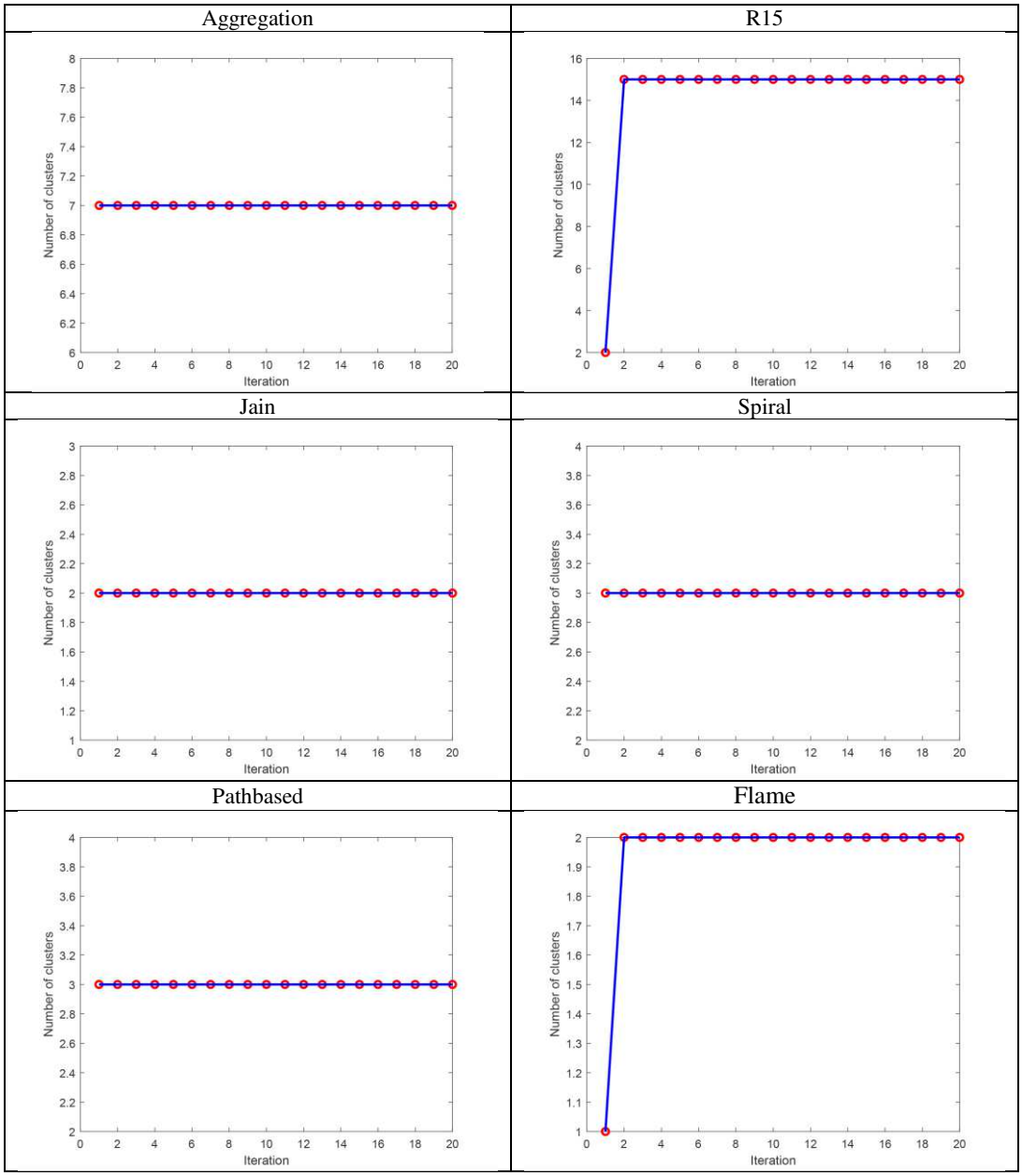

Fig. 11. Number of clusters estimated by GFO-DBSCAN in 20 iterations in synthetic data

4.3 convex data sets

These data sets include Data-2, Data-3, Data-4, Data-5, S1, S2 that have two dimensions. Hyper-spherical and hyper-elliptic clusters are the main features of these datasets. Data-2 to Data-4 have fewer objects and clusters compared to S1, S2 which these features have made it difficult to reach the optimal number of clusters in S1 and S2. According to Fig. 12, each cluster is marked with different colors. Each data set is clustered by GFO-DBSCAN in two dimensions that outliers are highlighted in red. The outliers lead to better separation of clusters and make that better identified to reach the best number of clusters in automatic clustering. The results of automatic clustering by GFO-DBSCAN has been shown in Table 6. The optimal values of esp and $minpts$ have been shown in the second and third columns respectively. GFO-DBSCAN has obtained the optimum number of clusters in all data sets according to the fourth column and the value of best costs that has been presented in the fifth column. In Fig.13, the number of clusters has been shown that GFO-DBSCAN has obtained in 20 iterations in convex data. GFO-DBSCAN has had superior performance in Data-2 and Data-3 according to Fig.13. In Data-4 and Data-5, the best number of clusters has been identified after a few iterations. Despite a large number of clusters in S1 and S2, the best number of clusters has been identified after passing a few iterations.

Table 6. the performance of GFO-DBSCAN in convex data sets.

Data	The optimal value of eps	The optimal value of $minpts$	Number of the best cluster	Best costs	Number of the records
Data-2	0.9219	19	2	-280.3001	200
Data-3	1.5518	17	3	-86.3073	250
Data-4	0.9203	4	4	-4485.3451	400
Data-5	0.7921	12	5	-2781.585	500
S1	1.8394	13	15	-627.9109	5000
S2	4.2040	71	15	-1092.0829	5000

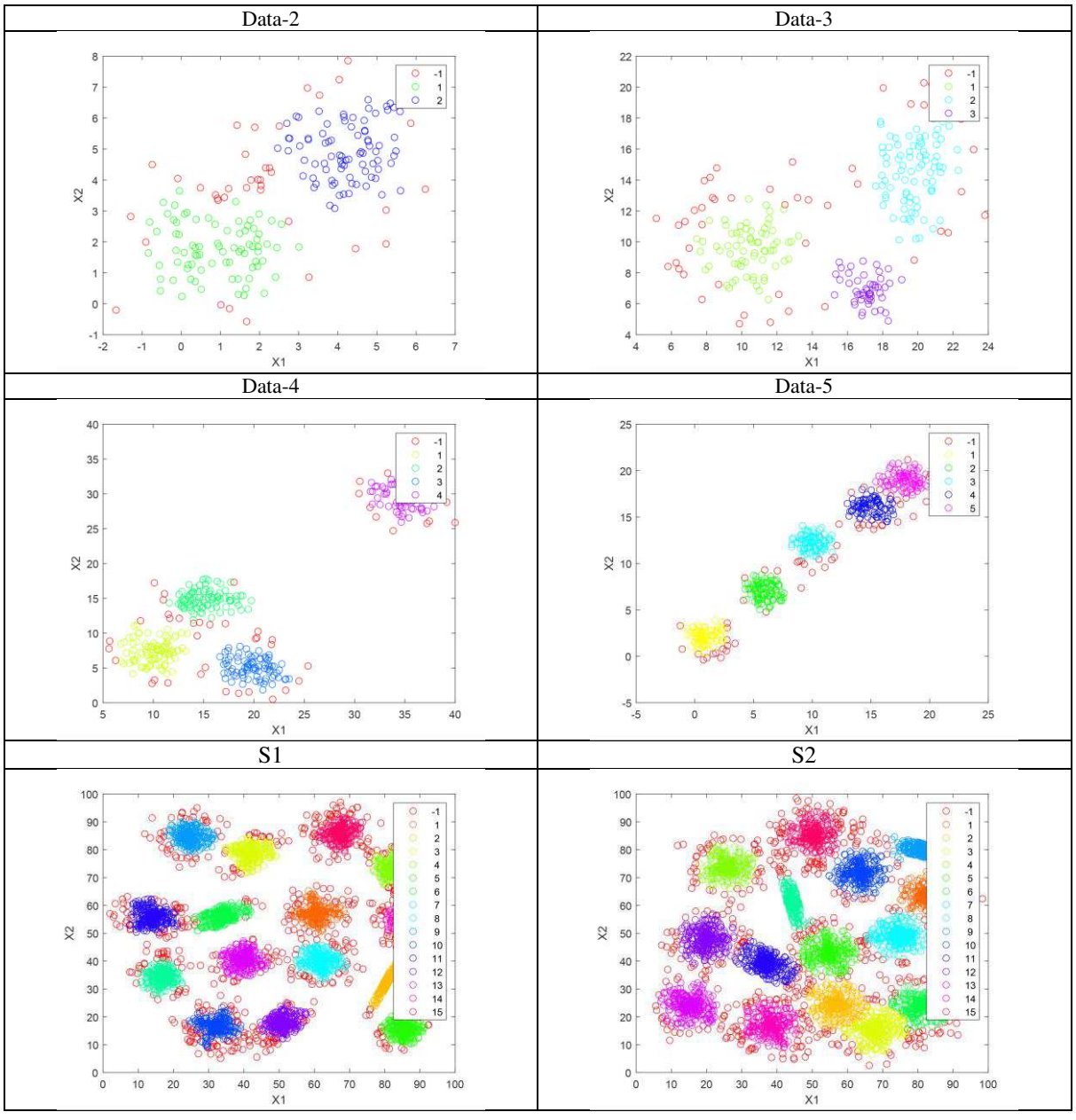

Fig.12. Automatic clustering by GFO-DBSCAN in convex data sets.

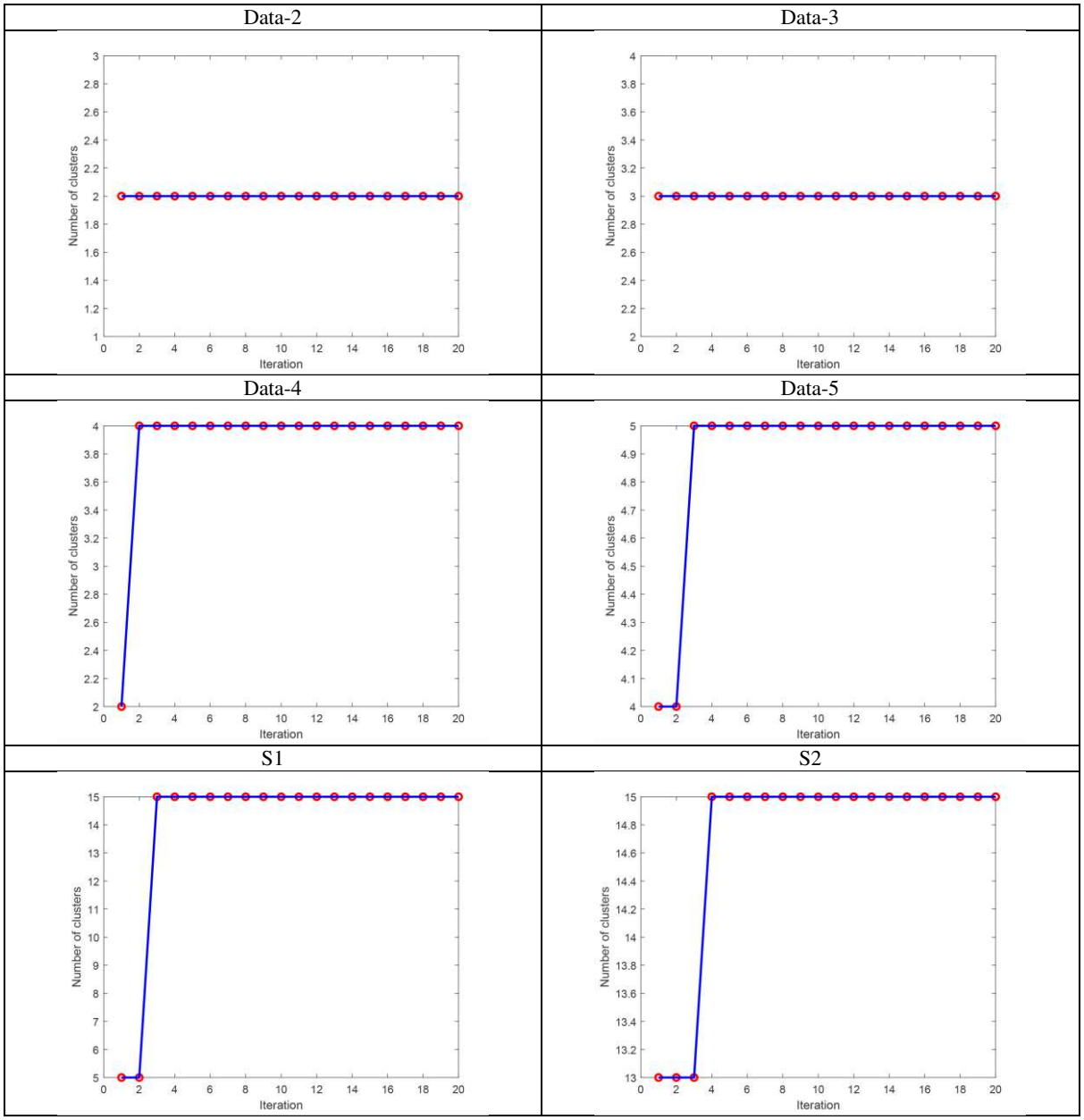

Fig. 13. Number of clusters estimated by GFO-DBSCAN in 20 iterations in convex data

4.4 real data sets

Real data sets are proper tools for investigating the performance of automatic clustering algorithms. The objects of these data sets have been often collected based on real experiments of the real problems. The dimension of these data sets are often more than two and has always been a challenge for automatic clustering algorithms because of their special structure. In this paper, four real data sets have been used to investigate the performance of GFO-DBSCAN. This perspective view of these data has been plotted according to Fig. 14. These data sets include Iris, Wine, Breast-Cancer, and Vowel. Iris has three optimal clusters and 150 objects that have been investigated in four dimensions. The Wine data has three optimal clusters,178 points, and five dimensions. Breast-Cancer has two clusters and 699 points which have been investigated in nine dimensions. The vowel data includes 5678 points and nine dimensions which have six optimal clusters. Four well-known automatic clustering algorithms have been used to compare the comparative results. These algorithms include DCPSO [49], ACDE [50], CGUK [11]. The results of these algorithms are based on [14] in which the ACDE was compared with other well-known algorithms. The comparative results have been presented in Table 7. The average values and variances in 40 times iteration have been shown in Tabel 7. The results of Iris show that GFO-DBSCAN has had proper

performance compared to other algorithms and has achieved the best result. The best result in Wine has belonged to DCPSO while there is very little difference between the results. In Breast-Cancer, the GFO-DBSCAN and ACDE have had the best results and have achieved the best number of clusters in all iterations. It is obvious that GFO-DBSCAN has found the optimal number of clusters in most of iterations and has achieve the best results in Vowel data.

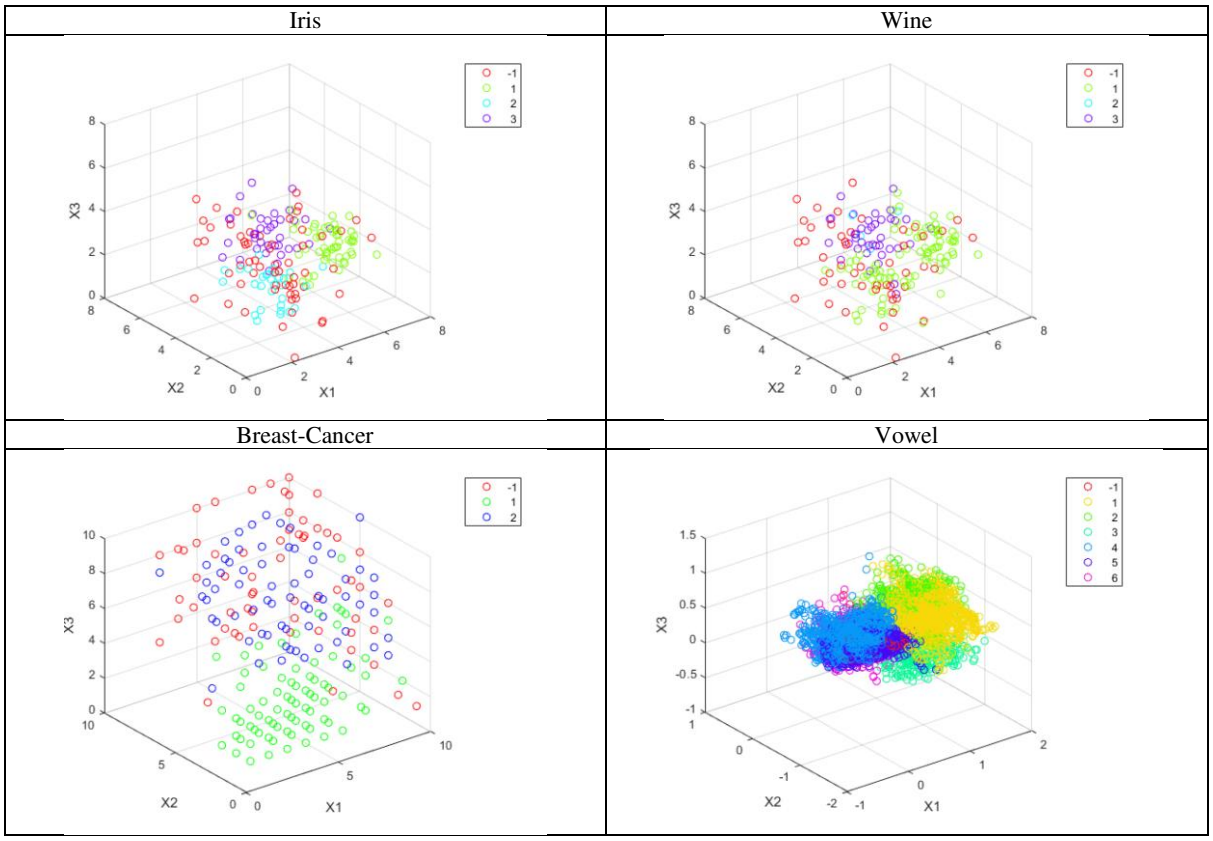

Fig. 14. Data plot for real data

Table 7. The comparative results between GFO-DBSCAN and other automatic clustering algorithms

Dataset	The actual number of cluster	GFO-DBSCAN	ACDE	DCPSO	GCUK
Iris	3	**2.92±0.0711**	3.25±0.0382	2.23±0.0443	2.35±0.0985
Wine	3	2.87±0.1121	3.25±0.0391	**3.05±0.0352**	2.95±0.0112
Breast-Cancer	2	**2.00±0.00**	**2.00±0.00**	2.25±0.0632	2.00±0.0083
Vowel	6	**5.82±0.1480**	5.75±0.0751	7.25±0.0183	5.05±0.0075

5. Conclusion

In this paper. A new automatic clustering called GFO-DBSCAN was proposed that was able to detect the best number of clusters. GFO-DBSCAN had two steps, in the first step, the grouper fish – octopus algorithm (GFO) algorithm was used for generating values of eps and $minpts$ that are the inputs of DBSCAN algorithm. In the second step, using eps and $minpts$, the clustering was done by DBSCAN algorithm and then the quality of the clustering was calculated by CH validity index. The different values of eps and $minpts$ made different results in clustering by DBSCAN that led to different results in CH index. The experimental results were divided into two sections. In the first section, GFO was investigated by some well-known benchmark functions and compared to four famous metaheuristic algorithms. The experimental results of GFO showed that there was a good balance between exploration and exploitation. Also, a new crossover operation was introduced in GFO that had a big

effect on the performance of GFO. Three categories of data were used to evaluate the efficiency of GFO-DBSCAN. The first category of data sets included synthetic data sets, convex data was the second category and the third category was real data sets. The proper performance of GFO-DBSCAN was quite obvious in the synthetic data sets. GFO-DBSCAN was able to detect the optimal number of clusters in all synthetic data especially in data with the non-convex clusters. In the convex data sets, GFO-DBSCAN was superior performance and in all data, the optimal number of the cluster was detected. In real data sets, GFO-DBSCAN was compared with three well-known automatic clustering algorithms. The comparative results showed that GFO-DBSCAN was achieved the optimal number of clusters in most data sets. The results of GFO-DBSCAN had two main reasons. Detecting the non-convex clusters by DBSCAN and detecting and removing outliers in process of GFO-DBSCAN algorithm were two main reasons for the algorithm's success in automatic clustering.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Armano, G. and M.R. Farmani, *Multiobjective clustering analysis using particle swarm optimization.* Expert Systems with Applications, 2016. **55**: p. 184-193.
2. Balavand, A., A.H. Kashan, and A. Saghaei, *Automatic clustering based on crow search algorithm-Kmeans (CSA-Kmeans) and data envelopment analysis (DEA).* International Journal of computational intelligence systems, 2018. **11**(1): p. 1322-1337.
3. Lloyd, S., *Least squares quantization in PCM.* IEEE transactions on information theory, 1982. **28**(2): p. 129-137.
4. Ester, M., et al. *A density-based algorithm for discovering clusters in large spatial databases with noise*. in *Kdd*. 1996.
5. José-García, A. and W. Gómez-Flores, *Automatic clustering using nature-inspired metaheuristics: A survey.* Applied Soft Computing, 2016. **41**: p. 192-213.
6. Omran, M.G., A. Salman, and A.P. Engelbrecht, *Dynamic clustering using particle swarm optimization with application in image segmentation.* Pattern Analysis and Applications, 2006. **8**(4): p. 332-344.
7. Das, S., A. Abraham, and A. Konar. *Spatial information based image segmentation using a modified particle swarm optimization algorithm*. in *Sixth International Conference on Intelligent Systems Design and Applications*. 2006. IEEE.
8. Maulik, U. and S. Bandyopadhyay, *Fuzzy partitioning using a real-coded variable-length genetic algorithm for pixel classification.* IEEE Transactions on geoscience and remote sensing, 2003. **41**(5): p. 1075-1081.
9. Bezdek, J.C., *Pattern recognition with fuzzy objective function algorithms*. 2013: Springer Science & Business Media.
10. Das, S., A. Abraham, and A. Konar, *Automatic kernel clustering with a multi-elitist particle swarm optimization algorithm.* Pattern recognition letters, 2008. **29**(5): p. 688-699.
11. Bandyopadhyay, S. and U. Maulik, *Genetic clustering for automatic evolution of clusters and application to image classification.* Pattern recognition, 2002. **35**(6): p. 1197-1208.
12. Qu, J., Z. Shao, and X. Liu, *Mixed PSO clustering algorithm using point symmetry distance.* Journal of Computational Information Systems, 2010. **6**(6): p. 2027-2035.
13. Dorigo, M., V. Maniezzo, and A. Colorni, *Ant system: optimization by a colony of cooperating agents.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 1996. **26**(1): p. 29-41.
14. Kanade, P.M. and L.O. Hall. *Fuzzy ants as a clustering concept*. in *22nd International Conference of the North American Fuzzy Information Processing Society, NAFIPS 2003*. 2003. IEEE.
15. Handl, J., J. Knowles, and M. Dorigo, *Ant-based clustering and topographic mapping.* Artificial life, 2006. **12**(1): p. 35-62.
16. Mehrabian, A.R. and C. Lucas, *A novel numerical optimization algorithm inspired from weed colonization.* Ecological informatics, 2006. **1**(4): p. 355-366.
17. Chowdhury, A., S. Bose, and S. Das. *Automatic clustering based on invasive weed optimization algorithm*. in *International Conference on Swarm, Evolutionary, and Memetic Computing*. 2011. Springer.
18. Bandyopadhyay, S. and S. Saha, *A point symmetry-based clustering technique for automatic evolution of clusters.* IEEE Transactions on Knowledge and Data Engineering, 2008. **20**(11): p. 1441-1457.
19. Sheng, W., et al., *A weighted sum validity function for clustering with a hybrid niching genetic algorithm.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2005. **35**(6): p. 1156-1167.
20. Zhang, C., D. Ouyang, and J. Ning, *An artificial bee colony approach for clustering.* Expert systems with applications, 2010. **37**(7): p. 4761-4767.
21. Kuo, R.-J., et al., *Automatic kernel clustering with bee colony optimization algorithm.* Information Sciences, 2014. **283**: p. 107-122.

22. Arya, R. and G. Sikka. *An Optimized Approach for Density Based Spatial Clustering Application with Noise*. in *ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India-Vol I*. 2014. Springer.
23. Dharni, C. and M. Bnasal. *An improvement of DBSCAN Algorithm to analyze cluster for large datasets*. in *2013 IEEE international conference in MOOC, innovation and technology in education (MITE)*. 2013. IEEE.
24. Smiti, A. and Z. Elouedi. *Dbscan-gm: An improved clustering method based on gaussian means and dbscan techniques*. in *2012 IEEE 16th international conference on intelligent engineering systems (INES)*. 2012. IEEE.
25. Darong, H. and W. Peng, *Grid-based DBSCAN algorithm with referential parameters.* Physics Procedia, 2012. **24**: p. 1166-1170.
26. Karami, A. and R. Johansson, *Choosing DBSCAN parameters automatically using differential evolution.* International Journal of Computer Applications, 2014. **91**(7): p. 1-11.
27. Jiang, H., et al., *A new hybrid method based on partitioning-based DBSCAN and ant clustering.* Expert Systems with Applications, 2011. **38**(8): p. 9373-9381.
28. Rad, M.H. and M. Abdolrazzagh-Nezhad, *A new hybridization of DBSCAN and fuzzy earthworm optimization algorithm for data cube clustering.* Soft Computing, 2020. **24**(20): p. 15529-15549.
29. Zhu, Q., X. Tang, and A. Elahi, *Application of the novel harmony search optimization algorithm for DBSCAN clustering*. Expert Systems with Applications, 2021. **178**: p. 115054.
30. Kim, M. and R. Ramakrishna, *New indices for cluster validity assessment.* Pattern Recognition Letters, 2005. **26**(15): p. 2353-2363.
31. Davies, D.L. and D.W. Bouldin, *A cluster separation measure.* IEEE transactions on pattern analysis and machine intelligence, 1979(2): p. 224-227.
32. Unsworth, R.K. and L.C. Cullen-Unsworth, *An inter-specific behavioural association between a highfin grouper (Epinephelus maculatus) and a reef octopus (Octopus cyanea).* Marine Biodiversity Records, 2012. **5**.
33. Olorunda, O. and A.P. Engelbrecht. *Measuring exploration/exploitation in particle swarms using swarm diversity*. in *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*. 2008. IEEE.
34. Alba, E. and B. Dorronsoro, *The exploration/exploitation tradeoff in dynamic cellular genetic algorithms.* IEEE transactions on evolutionary computation, 2005. **9**(2): p. 126-142.
35. Lin, L. and M. Gen, *Auto-tuning strategy for evolutionary algorithms: balancing between exploration and exploitation.* Soft Computing-A Fusion of Foundations, Methodologies and Applications, 2009. **13**(2): p. 157-168.
36. Khan, M.M.R., et al. *ADBSCAN: adaptive density-based spatial clustering of applications with noise for identifying clusters with varying densities*. in *2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT)*. 2018. IEEE.
37. Fielding, A., *Cluster and classification techniques for the biosciences*. 2007: Cambridge University Press Cambridge.
38. Xu, R. and D. Wunsch, *Clustering*. 2008, John Wiley & Sons.
39. Krause, E.F., *Taxicab geometry: An adventure in non-Euclidean geometry*. 2012: Courier Corporation.
40. Van Der Heijden, F., et al., *Classification, parameter estimation and state estimation: an engineering approach using MATLAB*. 2005: John Wiley & Sons.
41. Socha, K. and M. Dorigo, *Ant colony optimization for continuous domains.* European journal of operational research, 2008. **185**(3): p. 1155-1173.
42. Storn, R. and K. Price, *Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces.* Journal of global optimization, 1997. **11**(4): p. 341-359.
43. Eberhart, R. and J. Kennedy. *A new optimizer using particle swarm theory*. in *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*. 1995. IEEE.
44. Gionis, A., H. Mannila, and P. Tsaparas, *Clustering aggregation.* Acm transactions on knowledge discovery from data (tkdd), 2007. **1**(1): p. 4-es.
45. Veenman, C.J., M.J.T. Reinders, and E. Backer, *A maximum variance cluster algorithm.* IEEE Transactions on pattern analysis and machine intelligence, 2002. **24**(9): p. 1273-1280.
46. Jain, A.K. and M.H. Law. *Data clustering: A user's dilemma*. in *International conference on pattern recognition and machine intelligence*. 2005. Springer.
47. Chang, H. and D.-Y. Yeung, *Robust path-based spectral clustering.* Pattern Recognition, 2008. **41**(1): p. 191-203.
48. Fu, L. and E. Medico, *FLAME, a novel fuzzy clustering method for the analysis of DNA microarray data.* BMC bioinformatics, 2007. **8**(1): p. 1-15.
49. Omran, M.G., A. Salman, and A.P. Engelbrecht, *Dynamic clustering using particle swarm optimization with application in image segmentation.* Pattern Analysis and Applications, 2006. **8**(4): p. 332.
50. Das, S., A. Abraham, and A. Konar, *Automatic clustering using an improved differential evolution algorithm.* IEEE Transactions on systems, man, and cybernetics-Part A: Systems and Humans, 2008. **38**(1): p. 218-237.