JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 37, 157-183 (2021)
DOI: 10.6688/JISE.202101_37(1).0011

Determining the Parameters of DBSCAN Automatically Using the Multi-Objective Genetic Algorithm

ZEINAB FALAHIAZAR[1], ALIREZA BAGHERI[2] AND MIDIA RESHADI[3]
[1,3]*Department of Computer Engineering*
Science and Research Branch, Islamic Azad University
Tehran, Iran
E-mail: {zfalahiazar; reshadi}@srbiau.ac.ir
[2]*Department of Computer Engineering*
Amirkabir University of Technology
Tehran, Iran
E-mail: ar_bagheri@aut.ac.ir

Amongst clustering algorithms, density-based clustering has many advantages, including simplicity and the ability to detect clusters of different shapes and discover outliers. Nevertheless, all current density-based clustering algorithms need input parameters. These are difficult to determine and have a substantial impact on the clustering results. DBSCAN is a density-based clustering algorithm that has been widely used in different scientific fields for many years. In this study, a hybrid algorithm named MOGA-DBSCAN, which combines the DBSCAN algorithm with the multi-objective genetic algorithm (MOGA), has been proposed. In this algorithm, the clustering problem is regarded as a multi-objective optimization problem to optimize certain cluster validity indices, which indicate the goodness of the clustering solutions. In this way, the appropriate values of the DBSCAN parameters could be determined automatically. NSGA-II is used to solve this optimization problem, and a new cluster validity index based on detected outliers is used as a fitness function to increase the quality of solutions. The use of such a multi-objective algorithm optimizes several indices simultaneously and yields high-quality results. Moreover, the Delaunay triangulation algorithm, which needs no input parameters, is used to determine the initial bounds of the DBSCAN parameters to reduce the number of generations. The results indicate that the proposed algorithm has an acceptable level of accuracy in determining the DBSCAN parameters.

Keywords: density-based clustering, DBSCAN, multi-objective optimization, delaunay triangulation, cluster validity index

1. INTRODUCTION

Clustering algorithms are generally divided into five groups, including partitioning-based, hierarchical-based, density-based, model-based, and grid-based algorithms [1]. Density-based algorithms are one of the most widely used methods in various scientific fields [2-7]. These algorithms detect low-density and high-density areas and separate them from each other. DBSCAN is the main algorithm that belongs to this group. It needs the two input parameters, *Eps* and *MinPts*, and is able to detect outliers and clusters of different shapes. However, it is difficult to determine the input parameters of this algorithm, as its values considerably affect the clustering results. Furthermore, the algorithm cannot detect clusters correctly either when the border points of the two clusters

Received December 1, 2019; revised January 27, 2020; accepted February 26, 2020.
Communicated by Shyi-Ming Chen.

are close to each other or when there are multi-density clusters. Many solutions have been provided in the literature to resolve such issues. In [8], the OPTICS algorithm is introduced as an extension of the DBSCAN algorithm. This algorithm creates a cluster ordering to extract clusters automatically and interactively. Such an ordering indicates the density-based clustering structure of the data. In order to create a cluster order, two new distances are calculated for each point: 1- Core-distance, 2- Reachability-distance. The Core-distance of a point p is the smallest radius of the neighborhood that makes the p a core point. The Reachability-distance of a point q from a core point p is the maximum value of *Core-distance*(p) and *Euclidean-distance*(p, q). A cluster order is created based on the Reachability-distance of points. In this method, for a value of *MinPts* that is determined by the user, the value of the *Eps* parameter could be determined by using the obtained ordering. Nevertheless, it is still necessary for the users to determine the *MinPts* value, and the interaction with users is required to determine the *Eps* values. In some researches, density is studied locally to detect the clusters of different densities. However, this method increases the number of input parameters. In [9], the proposed LDBSCAN algorithm uses the two concepts of local outlier factor (LOF) and local reachability factor (LRF) to detect multi-density clusters. This algorithm needs four input parameters. In [10], the concept of the *k*-dist plot is utilized to determine the different densities. The *k*-dist plot curve is drawn by using the sorted distances of any point from its kth nearest neighbor. Any significant change in this curve determines a value for the *Eps* parameter. In another study, the KD-tree data structure is used for detecting clusters of different densities [11]. The *k*-distance curve is drawn based on this structure and a set of the *Eps* parameter values is estimated. It is necessary to determine the parameter *k* and interact with users in such methods as well. In [12], the grid partition technique is employed to automatically determine the input parameters of DBSCAN. However, other input parameters are necessary for grid partitioning. In [13], Gaussian mean is used for determining the DBSCAN parameters. This method also needs input parameters for Gaussian distribution. The authors of [14] proposed the LP-DBSCAN algorithm, which uses the Density Peak Clustering algorithm to determine the parameters of DBSCAN. This algorithm finds the peaks of density but requires a cut-off distance, which is determined by the user.

The heuristic algorithms described above are not fully automatic and require the user to specify at least one parameter. Recently, metaheuristic algorithms have been used in many studies to improve the clustering results [15-22]. However, few studies have employed such algorithms to determine the DBSCAN parameters automatically. In [23], the use of a hybrid method called BDE-DBSCAN, which combines DBSCAN with the binary differential evolution (BDE) algorithm that has a structure similar to the genetic algorithm, is proposed to determine the DBSCAN parameters. This hybrid method uses the external index of purity as the objective function. However, clusters are not known in advance in the case of real-world datasets. Hence, this method cannot be functional. In [24], the DBSCAN algorithm is used for clustering the data, whereas the genetic algorithm is employed to determine the parameters of DBSCAN automatically. The single-objective genetic algorithm based-DBSCAN is shown in Algorithm 1. In Step 1, the bounds of *Eps* are considered between the minimum and maximum distances of each point from its nearest neighbor. The bounds of *MinPts* are considered between one unit larger than the dimension of the input vector and half of the total number of data items. The Offspring Population is generated from executing crossover and mutation operators

in Step 4. Every chromosome comprises two genes, which indicate the values of the *Eps* and *MinPts* parameters. In Steps 2 and 5, first, the DBSCAN algorithm is run with values of every chromosome as *Eps* and *MinPts* parameters, then clustering results are scored with the objective function. The silhouette internal index is used as the objective function. In Step 6, the best solutions are selected by the select operator.

Algorithm 1: Pseudo-code of single-objective genetic algorithm-DBSCAN

```
    // P is output of Function //
    // P = Parent Population, Q = Offspring Population, S = Input Dataset //
Function SOGA-DBSCAN
1   Initialize the P randomly within the minimum and maximum bounds;
2   Run objective function over the results of DBSCAN(S, P);
3   Repeat
4       Q = Union(Mutation(P), Crossover(P));
5       Run objective function over the results of DBSCAN(S, Q));
6       [P] = Selection(Q, P);
7   Until stopping condition
End-Function
```

The above studies have used single-objective optimization to determine the parameters of DBSCAN. In [25], a multi-objective genetic algorithm is utilized for partition-based fuzzy clustering. In this clustering method, all possible partitions of the dataset and the values of the validity index specify the entire search space. Conventional partitioning methods, such as *K*-mean, utilize the greedy search technique in the search space to optimize the compactness of the clusters. However, although such algorithms are computationally effective, they often get trapped in some local optima depending on the selection of the initial cluster centers. Furthermore, they optimize one validity index and do not cover different features of the dataset. On the contrary, multi-objective clustering methods can optimize more than one validity index simultaneously. This helps in obtaining high-quality results. In this study, a multi-objective genetic algorithm is used to determine the parameters of DBSCAN automatically.

The contributions presented in this study include:

- MOGA is combined with DBSCAN to determine the parameters of DBSCAN automatically. Therefore, more than one index is considered during clustering, and it improves the quality of the clustering result. The internal cluster validity indices are used as objective functions.
- A new cluster validity index is proposed based on the outliers detected by DBSCAN to increase the diversity and quality of solutions.
- The Delaunay triangulation, which does not require input parameters, is used for determining the *Eps* and *MinPts* bounds. Specifying the bounds of parameters will reduce the number of iterations and thus reduce computational time.
- Various experiments are performed on standard data sets to check the performance of the proposed method.

The rest of the study is organized as follows: Section 2 describes the related concepts such as the DBSCAN algorithm, MOGA, the Delaunay triangulation, and the clus-

ter validity indices. Section 3 introduces the proposed algorithm for automatically determining DBSCAN parameters. The implementation results are evaluated in Section 4. Finally, Section 5 presents the conclusions.

2. RELATED CONCEPTS

This section describes the algorithms and concepts used in the proposed method to determine the DBSCAN parameters automatically.

2.1 DBSCAN Clustering

The DBSCAN algorithm relies on a density-based notion of the cluster and is the most prevalent density-based clustering algorithm [26]. This type of clustering, which has been designed to detect clusters of arbitrary shapes, is also capable of finding outliers. The DBSCAN algorithm has the following two parameters:

- *Eps*: the radius of the neighborhood around a point *x*
- *MinPts*: the minimum number of neighbors within the "*Eps*" radius

Some concepts of DBSCAN are defined as follows:
(1) Core point: a point that has at least the minimum number of points (*MinPts*) in the neighborhood radius (*Eps*).
(2) Directly density-reachable: a point *p* is directly density-reachable from a point *q* with regard to the parameters *Eps* and *MinPts* if *p* is within the neighborhood radius of *q*, and *q* is a core point.
(3) Density-reachable: a point *p* is density-reachable from a point *q* with regard to *Eps* and *MinPts* if there is a chain of points $p_1 ... p_n$ where $p_1 = q$ and $p_n = p$ such that p_{i+1} is directly density-reachable from p_i.

The DBSCAN algorithm comprises the following steps:
- Selection of an arbitrary point *p*.
- Retrieval of all points that are density-reachable from *p* with regard to *Eps* and *MinPts*.
- If *p* is a core point, the cluster is created.
- If *p* is not a core point, no points are density-reachable from *p*, and DBSCAN visits the next point in the database.
- This process is continued until all points are processed.

Points that are not included in clusters are considered outliers. The time complexity of DBSCAN is $O(n^2)$, where *n* is the number of points. If a spatial index is used, the time complexity will be $O(n\log n)$. If the *Eps* and *MinPts* parameters are adjusted appropriately, the algorithm will effectively determine clusters of arbitrary shapes.

2.2 The Multi-Objective Genetic Algorithm

The single-objective genetic algorithm optimizes only one objective in the search process and, eventually, produces one optimal solution. However, when solving many

real problems, there are a large number of contradictory objectives that must be optimized simultaneously. As a result, multi-objective optimization algorithms are employed to solve real problems. These algorithms provide a set of non-dominated solutions in the target space. This set of non-dominated solutions prepares valuable information about the problem such that, depending on the designer's or decision maker's needs, the best solution is selected in the end.

A formal definition of the multi-objective optimization problem is presented as follows [25, 27]:

$$\begin{aligned} &\bar{x} = [x_1, x_2, ..., x_n]^T \\ &g_i(\bar{x}) \geq 0, \quad i = 1, 2, ..., m \\ &h_i(\bar{x}) = 0, \quad i = 1, 2, ..., p \\ &\bar{f}(\bar{x}) = [f_1(\bar{x}), f_2(\bar{x}), ..., f_k(\bar{x})]^T \end{aligned} \tag{1}$$

A solution $\bar{x}$ is a vector of n decision variables, which satisfies the m inequality and the p equality constraints and optimizes the vector function as a maximization or minimization.

Based on the given constraints, a feasible space F that includes all acceptable solutions is defined. MOGA tries to promote the convergence of solutions toward the Pareto optimal front. After considering the optimization as a minimization problem, a decision vector x^* can be regarded as a part of the Pareto optimal front if and only if there is no vector $\bar{x}$ that can dominate x^*:

$$\begin{aligned} &\forall i = 1, 2, ...k, \; f_i(\bar{x}) \geq f_i(\bar{x}^*) \\ &\nexists i \in 1, 2, ...k, \; f_i(\bar{x}) < f_i(\bar{x}^*) \end{aligned} \tag{2}$$

In this study, the elitist non-dominated sorting GA (NSGA-II) that is a multi-objective evolutionary algorithm based on non-dominated sorting, is used to optimize the objectives. NSGA-II is a Pareto-based elitist algorithm that uses non-dominated sorting and crowding distance [28]. In this algorithm, the solutions are first sorted by using non-dominated sorting and then selecting them from the Pareto front. When the number of required solutions is smaller than the number of solutions in the Pareto front, solutions with the longest crowding distance are selected.

2.3 Delaunay Triangulation

In mathematics and calculus geometry, a Delaunay triangulation is denoted by $D(S)$ for a set S of points in a plane so that no point in S is inside the circumcircle of any triangle in $D(S)$ [29, 30]. This triangulation was developed by Boris Delaunay and is used in different applications of geographic information systems (GIS).

If there is a set of points $S = \{p_0, p_1, \ldots, p_{n-1}\}$ on a plane, the Voronoi region of the point $p_i \in S$ is a set of points in the R^2 where p_i is in their nearest neighbor.

$$\{x \in R^2 | \forall_{j \neq i}, \; d(x, p_i) \leq d(x, p_j)\} \tag{3}$$

In this equation, d is the distance function. The Voronoi diagram is formed by n Vo-

ronoi regions of S. These regions are convex polygons with separate inner spaces. Based on the Voronoi diagram, the Delaunay triangulation is defined as a planar graph as follows: the nodes of $D(S)$ include the data points of S and the two nodes p_i and p_j are connected by an edge if the borders of their Voronoi regions have a shared line. Fig. 1 shows the Delaunay triangulation of 15 points.

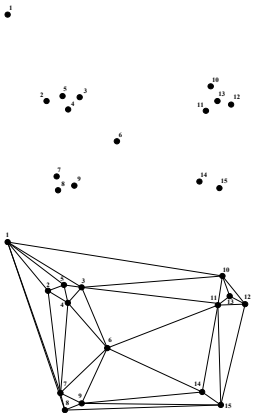


Fig. 1. Delaunay triangulation of 15 points [30].

2.4 Cluster Validity Indices

Basically, the following are the two types of cluster validity indices: external and internal. The external cluster validity indices are used for comparing the clustering results with accurate clustering of data. These indices are very useful for comparing the efficiency of different clustering methods when an accurate clustering is known. On the other hand, the internal cluster validity indices evaluate the quality of the clustering results with respect to the geometric features of clusters, such as compactness, separation, and connectedness.

(A) Internal Cluster Validity Indices

This section explains some of the internal cluster validity indices which are used in the experiments.

(1) Silhouette index [33]

Assume a_i is the average distance of a point x_i from the other points of the same cluster, and b_i is the minimum of the average distances of this point from the other clusters. Subsequently, the silhouette width of the point can be defined as follows:

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)}. \tag{4}$$

Silhouette index is the average silhouette width of all the data points.

$$S = \frac{1}{n}\sum_{i=1}^{n} s_i \tag{5}$$

The value of the Silhouette index changes between –1 and 1, and the higher values indicate better clustering results.

(2) Dunn index [34]

Suppose $\delta(C_i, C_j)$ shows the distance between the two clusters C_i and C_j, and $\Delta(C_i)$ shows the diameter of the cluster C_i. The Dunn index is defined as follows:

$$DN = \min_{1\le i\le K}\left\{\min_{1\le j\le K, j\ne i}\left\{\frac{\delta(C_i, C_j)}{\max_{1\le k\le K}\{\Delta(C_k)\}}\right\}\right\}. \tag{6}$$

In this index, δ and Δ are defined as follows:

$$\begin{aligned}&\delta(C_i, C_j) = \min_{x\in C_i, y\in C_j}\{D(x, y)\},\\ &\Delta(C_i) = \max_{x,y\in C_i}\{D(x, y)\}.\end{aligned} \tag{7}$$

Here, $D(x, y)$ shows the distance between the points x and y. The larger value of the Dunn index indicates better compaction and better cluster separation. Thus, the goal is to maximize the Dunn index value.

(B) External Cluster Validity Indices

T is assumed to be the accurate clustering of a dataset, and C is assumed to be a clustering result that is obtained by a clustering algorithm. In addition, a, b, c, and d are the respective parameters that show the pairs of points depending on the same cluster in both T and C, the number of pairs depending on the same cluster in T but on different clusters in C, the number of pairs depending on different clusters in T but on the same cluster in C, and the number of pairs depending on different clusters in both T and C. Therefore, the six external validity indices can be defined as follows [25, 35]:

(1) Jaccard

The value of the Jaccard index changes between 0 and 1. The greater value of this index demonstrates the greater similarity between C and T. Therefore, $J(T, T) = 1$.

$$J(T, C) = \frac{a}{a+b+c} \tag{8}$$

(2) Rand

The value of this index is also between 0 and 1. The greater value of the Rand index indicates the greater similarity between T and C and $RI(T, T) = 1$. The Rand index, when multiplied by 100, is also known as the percentage of correctly classified pairs.

$$RI(T, C) = \frac{a+d}{a+b+c+d} \tag{9}$$

(3) Minkowski

The lower value of this index shows the greater similarity between T and C, such that $M(T, T) = 0$.

$$M(T, C) = \sqrt{\frac{b+c}{a+b}} \tag{10}$$

(4) F-measure

F-measure is the weighted harmonic mean of Precision (P) and Recall (R).

$$P(T,C) = \frac{a}{a+c} \tag{11}$$

$$R(T,C) = \frac{a}{a+c} \tag{12}$$

$$F = \frac{(\beta^2+1)PR}{\beta^2 P + R} \quad \text{where} \quad \beta[\ ,\infty] \tag{13}$$

If $\beta = 1$, then F is simplified as follows:

$$F_1 = \frac{2PR}{P+R} \tag{14}$$

The value of F_1 is between 0 and 1. The greater value of F_1 indicates the better similarity between T and C and $F_1(T, T) = 1$.

3. MOGA-DBSCAN: MULTI OBJECTIVE GENETIC ALGORITHM-BASED DBSCAN

In MOGA-DBSCAN, the DBSCAN clustering problem is considered as a multi-objective optimization problem. MOGA is used for this purpose such that every chromosome comprises two genes, thereby indicating the values of the *Eps* and *MinPts* parameters. Fig. 2 shows the flowchart of the proposed algorithm.

First, it is necessary to determine the *Eps* and *MinPts* bounds in MOGA appropriately to generate high-quality solutions with a smaller number of generations. In this study, the Delaunay triangulation is employed to determine the *Eps* and *MinPts* bounds. The advantage of this method over a KNN algorithm is that it needs no input parameters, and the nearest neighbor graph is a subgraph of the Delaunay triangulation. Moreover, it is computationally effective.

The *Eps* and *MinPts* bounds are determined as follows by using the Delaunay triangulation:

1. The length of the shortest edge < *Eps* < the average length of edges
2. For every point, the number of neighbors in a radius equal to the average length of edges is obtained, and, subsequently, the minimum and maximum numbers of neighbors are regarded as the *MinPts* bounds.

Algorithm 2 shows the steps in the MOGA-DBSCAN. In Step 5, the mutation and crossover operators are used for generating new solutions. These solutions are generated to optimize the objectives. The objectives to be optimized might express various features of the clusters, such as compactness, separation, and connectedness. The simultaneous optimization of multiple indices is more useful for attaining different features. This is because no cluster validity index operates in the same way for various types of datasets.

As there is no need to know the accurate clustering in the internal cluster validity indices, they are used as the objective functions of MOGA. In each generation, the DBSCAN algorithm is run with new solutions. Subsequently, every solution vector includes the two DBSCAN parameters. Then the clustering result is evaluated by the objective functions (Step 7). The different values of the DBSCAN parameters and the validity indices form the search space.

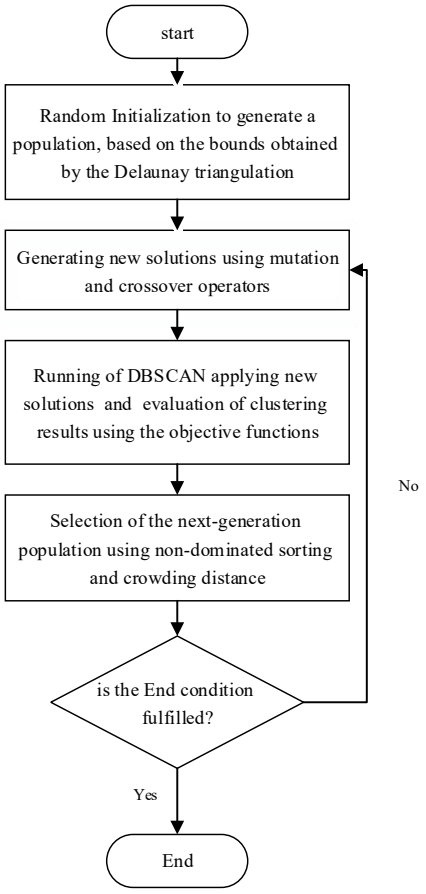


Fig. 2. The flowchart of MOGA-DBSCAN.

Algorithm 2: Pseudo-code of MOGA-DBSCAN

```
    // P and PF are outputs of Function //
    // P = Parent Population, Q = Offspring Population, PF = Pareto Front //
Function MOGA-DBSCAN
1   Initialize the P randomly within the minimum and maximum bounds;
2   Old_PF = NULL; // Old_PF = Old Pareto Front //
3   Run objective function1 and objective function2 over the results of DBSCAN(S, P));
4   For i = 1 to Max_Generation
5       Q = Union(Mutation(P), Crossover(P));
```

```
6       // S = Input Dataset//
7       Run objective function1 and objective function2 over the results of DBSCAN(S, Q) ;
8       [P PF] = Selection(Q, P);
9       If Old_PF is Empty
10            Old_PF = PF;
11      End-if
12      If (i mod n) == 0 // n = the interval between tow generation to check stopping
          conditions
13        Hypothesis1 = T-test(PF.f1,old_PF.f1); // f1= fitness of objective function1 //
14        Hypothesis2 = T-test(PF.f2,old_PF.f2); // f2= fitness of objective function 2 //
15        If   (Hypothesis1 = null hypothesis) And (Hypothesis2 = null hypothesis)
16              Break;
17        Else
18              Old_PF = PF;
19        End-if
20      End-if
21 End-For
22 End-Function
```

At the end of MOGA run time, the final set of the near-Pareto-optimal solutions includes a number of non-dominated solutions (Step 8). The users can select the appropriate solutions (DBSCAN parameters) depending on the requirements of the problem.

Furthermore, the *t*-test is used to decrease the number of iterations and terminate MOGA. In this statistical test, the results of two algorithms and the significance level (α) are regarded as inputs [36, 37]. The significance level indicates the sensitivity percentage of the *t*-test, which is usually equal to 0.05 by default. After applying the inputs to the *t*-test algorithm, the null hypothesis is either rejected or accepted by the algorithm. If the null hypothesis is accepted, then it means that the average outputs of the two optimization algorithms are the same (that is, the difference is not significant). In other words, the similarity between the outputs of the first and the second optimization algorithms is more than 5%. If the null hypothesis is rejected (the alternative hypothesis is accepted), it indicates that the average outputs of the two optimization algorithms are not the same (the difference is significant). In other words, the similarity between the outputs of the first and the second optimization algorithms is less than 5%.

In this study, the *t*-test is used in the following two cases: (1) terminating MOGA: after each *n* generation, the current solution is compared with the solution of n previous generations by using the *t*-test (steps 12 to 19). If the null hypothesis is accepted, then the algorithm gets terminated. If the null hypothesis is rejected and the similarity is less than 5%, then the algorithm keeps running; (2) comparing the results of both of the evaluated algorithms.

3.1 The Proposed Validity Index

The proposed index is based on the outliers detected by a clustering algorithm. The new outlier-based index is defined as follow:

$$os_i = min_{1 \le j \le m}(d_{ij})$$

$$Outlier-index=\frac{\sum os_i}{n}. \tag{15}$$

Where d_{ij} is the distance of the outlier i from cluster j, n is the number of outliers, and m is the number of clusters. Outlier-index is the average of the minimum distance of outliers to the clusters. This index shows the similarity of the detected outliers to the clustered points. The higher value of the Outlier-index shows that the detected outliers have a greater distance and less similarity to clustered points, and the outliers are more appropriately detected.

Fig. 3 illustrates Outlier-index with an example. In the first case, the value of *MinPts* is set to 2, and the value of *Eps* is set to 1.1. Therefore, two clusters are detected, and points 1 and 6 are detected as the outliers. If the value of *Eps* is changed to 1.6, two clusters are detected and only point 1 is detected as an outlier.

The value of Outlier-index is 3.2 in the first case and 4.5 in the second case so in terms of this index, the second clustering is better. The value of the Silhouette index is 0.83 and 0.79 in the first and second cases, respectively. If point 1 is actually a member of the cluster, the Outlier-index has done a better assessment. But if point 1 is an outlier, the Silhouette index has made a better evaluation of the clustering result.

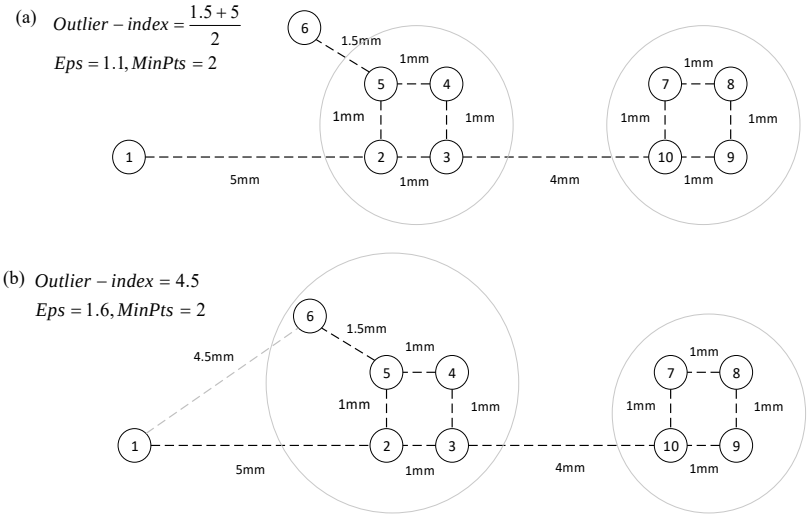


Fig. 3. Outlier-index on an example dataset.

In the case of real data that there is no previous knowledge about them, the simultaneous use of these two indices as contradictory objectives in the multi-objective optimization will increase the likelihood that there will be correct clustering among the set of solutions.

3.2 Time Complexity

The time complexity of MOGA-DBSCAN depends on the time complexities of DBSCAN, NSGA-II, and the Delaunay triangulation:

$$O(O(n\log n) + g \times (O(MN^2) + N \times (O(fitness) + O(n\log n)))) \quad (16)$$

Where g is the number of generations, and $O(fitness)$ indicates the time complexities of the cluster validity indices that are used as objective functions. The time complexity of DBSCAN and the Delaunay triangulation is $O(n\log n)$, where n is the total number of data items. The time complexity of NSGA-II is $O(MN^2)$ regardless of the objective function [28]. M indicates the number of objectives, whereas N is the population size. As the number of objectives and the population size is much smaller than the total number of data items, the time complexity of NSGA-II is $O(1)$. Therefore, the time complexity of MOGA-DBSCAN can be expressed as follows:

$$O(g \times ((fitness) + O(n\log n))). \quad (17)$$

The time complexity of Silhouette and Dunn is $O(n^2)$, where n is the total number of data items. The time complexity of the outlier-index is $O(nm)$, where m is the number of detected outliers.

4. IMPLEMENTATION AND EVALUATION

In this study, as explained in Section 2.2, NSGA-II is used as a non-dominated, sorting-based, multi-objective evolutionary algorithm to optimize objectives. Moreover, Outlier-index, Dunn and Silhouette are used as objective functions. When DBSCAN detects no outliers, the Outlier-index value is assumed to be the value of the Silhouette index instead of zero. Other indices can also be used as objective functions [25]. Dunn and Silhouette are employed for the following two reasons: (1) they try to optimize the compactness and separation features of clusters; and (2) they do not need to calculate the cluster center in contrast with many partitioning-based clustering indices.

In addition to the proposed algorithm, the method introduced in [24], which uses a single-objective genetic algorithm for determining the DBSCAN parameters, is implemented for the evaluation. The time complexity of the single-objective genetic algorithm depends on the time complexities of the mutation, the crossover, and the selection operators and on the objective function:

$$O(O(p_m \times N) + (O(p_n \times N) + O(N\log N) + N\, O(fitness)). \quad (18)$$

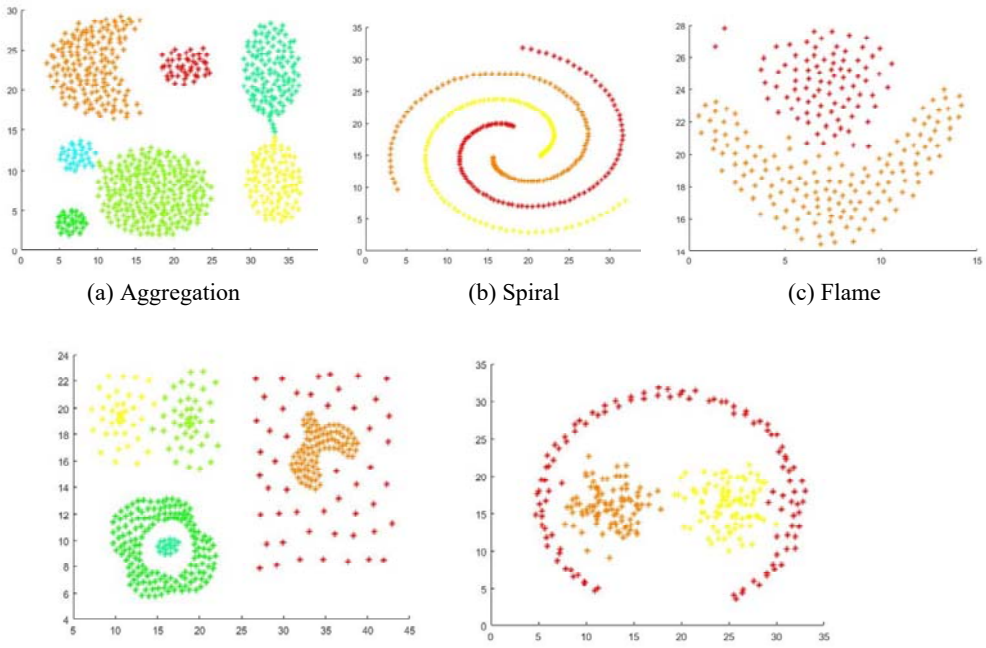
The coefficients p_m and p_n are the numbers that indicate the probabilities of mutation and crossover, respectively, and N is the population size. As the objective function run time is usually much longer than that of the genetic algorithm operators, the time complexity of the single-objective genetic algorithm is $O(fitness)$ [38]. In this study, the single-objective genetic algorithm is run once by utilizing the Silhouette index as an objective function and once by using the Dunn index as an objective function. As a result, $O(gn^2)$ is the time complexity of this algorithm in which n indicates the number of data items and g is used to signify the number of generations.

Fig. 4 shows the datasets that are used for evaluation, and Table 1 shows their characteristics. All implementations are carried out in MATLAB software, and all the experiments are run on a computer with a 3.7 GHz Core i3 processor and 8 GB RAM.

Table 1. Standard datasets [39].

Data set	Multi-density	Classes	Size
Aggregation	No	7	788
Spiral	Yes	3	312
Flame	No	2	240
Compound	Yes	6	399
Path-Based	No	3	300

Six external indices, namely, Jaccard, Rand, Minkowski, Precision, Recall, and F_1, are also used to evaluate the performance of the proposed algorithm. Since the upper bound of the Minkowski index is unclear, the relative difference is used for comparing the results in this study.



(a) Aggregation (b) Spiral (c) Flame

(d) Compound (e) Path-Based

Fig. 4. The five applied data sets.

4.1 Analysis

In this section, the results of running MOGA-DBSCAN and the single-objective genetic algorithm-based DBSCAN (SOGA-DBSCAN) are compared with the best result of the DBSCAN algorithm that is obtained with the help of the OPTICS algorithm by testing various *MinPts* values for each data set. In other words, the value of *Eps* can be determined for each *MinPts* value by using the trial and error method to find the best result.

Each algorithm is run 30 times on any dataset. In each run, the best solution is selected according to the six external indices. In all experiments, the termination conditions are checked after every 5 generations. Fig. 5 shows the average number of generations in 30 runs of MOGA-DBSCAN using Outlier-index and Silhouette as the objective functions for each dataset. Results of the experiments show that the use of the Delaunay triangulation to determine the bounds of *Eps* and *MinPts* reduces the number of generations compared to the method used in [24], which determines only bounds of *Eps*.

The *t*-test is used as a statistical test to examine the comparison accuracy of the two evaluated algorithms. The following tables indicate the accepted hypothesis (H_0 or H_1) for comparing the results of each pair of optimization algorithms based on the *t*-test. The significance level is 0.05 by default, which is the same for all simulations in this study. Tables 2-11 indicate the accepted hypothesis by the *t*-test related to the comparison of the evaluated algorithms and the average values of the indices. In general, if the *t*-test accepts the H_1 hypothesis for the comparison of two algorithms, the algorithm with a better average value in the validity index will produce better results in clustering. If the null hypothesis is accepted, then the two algorithms exhibit the same clustering operation regarding the validity index.

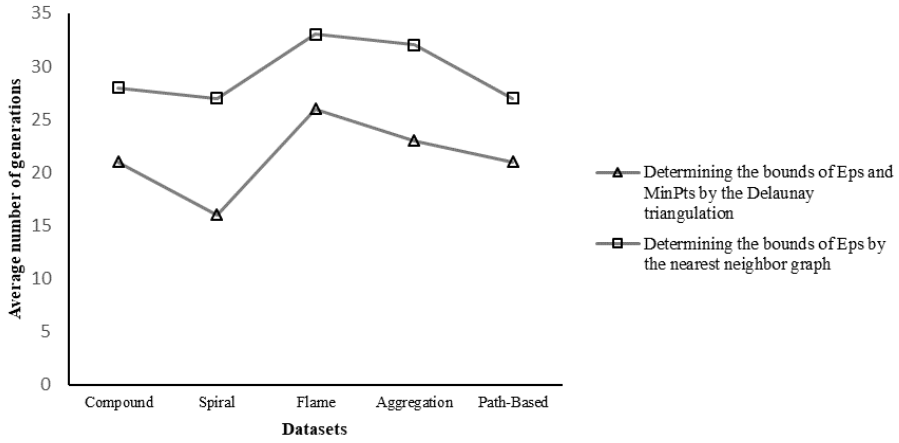


Fig. 5. The average number of generations in 30 runs of MOGA-DBSCAN.

According to Table 2, the *t*-test accepts the null hypothesis for the results of SOGA-DBSCAN in the compound dataset by employing Dunn and Silhouette as the objective functions separately. However, the *t*-test rejects the null hypothesis for the other comparisons.

According to Table 3, the best clustering results of the DBSCAN algorithm in the compound dataset are 5%, 1%, 34%, 3%, 5%, and 4% better than the results of MOGA-DBSCAN using Silhouette and Outlier-index as the objective functions for the Jaccard, Rand, Minkowski, Precision, Recall, and F_1 indices, respectively. Considering these indices, the results of MOGA-DBSCAN using Silhouette and Outlier-index as the objective functions are 21%, 8%, 53%, 18%, 19%, and 18% percent better than the results of MOGA-DBSCAN using Dunn and Silhouette. With respect to the Jaccard, Rand, Min-

kowski, Precision, Recall, and F_1 indices, the results of MOGA-DBSCAN using Dunn and Silhouette are 13%, 15%, 31%, 14%, 14%, and 14% better than those of SOGA-DBSCAN employing Silhouette as the objective function. They are also 20%, 20%, 38%, 18%, 18%, and 18% better than those of SOGA-DBSCAN using Dunn as the objective function, respectively. Therefore, MOGA-DBSCAN performed clustering better than SOGA-DBSCAN for the six validity indices, and SOGA-DBSCAN has the same clustering results using Dunn or Silhouette as the objective functions.

In the spiral dataset, the *t*-test accepts the null hypothesis for the results of SOGA-DBSCAN by employing Silhouette as the objective function and MOGA-DBSCAN using Dunn and Silhouette as the objective functions (Table 4). Therefore, MOGA-DBSCAN using Dunn and Silhouette as the objective functions, presents the same clustering as SOGA-DBSCAN using Silhouette as the objective function for the six indices and, subsequently, both algorithms presented better clustering results than SOGA-DBSCAN by employing Dunn as the objective function (Table 5). Also, the best clustering results of the DBSCAN algorithm in the spiral dataset are the same as the results of MOGA-DBSCAN using Silhouette and Outlier-index as the objective functions for the Jaccard, Rand, Minkowski, Precision, Recall, and F_1 indices, respectively.

Based on the results of the experiments in the flame dataset (Table 6), the t-test accepts the H_1 hypothesis for all comparisons in the flame dataset. Therefore, the results of all algorithms are comparable. Table 7 shows that MOGA-DBSCAN presents better clustering results than SOGA-DBSCAN using Dunn or Silhouette as the objective functions for the six validity indices. In addition, the results of MOGA-DBSCAN using Silhouette and Outlier-index as the objective functions are better than the results of MOGA-DBSCAN using Dunn and Silhouette as the objective functions.

The *t*-test accepts the null hypothesis for the best clustering results of the DBSCAN algorithm and MOGA-DBSCAN using Silhouette and Outlier-index as the objective functions in the Aggregation dataset in Table 8. Therefore, based on the six validity indices, MOGA-DBSCAN using Silhouette and Outlier-index as the objective function presents the same clustering as the best clustering results of the DBSCAN algorithm in Table 9.

Also, the results obtained from the experiments in the path-based dataset are similar to those obtained from the compound dataset in Table 10. With respect to the six validity indices, MOGA-DBSCAN performs clustering better than SOGA-DBSCAN in Table 11.

According to the results of the experiments, the following general conclusions can be derived:

(1) With respect to the Jaccard, Rand, Minkowski, Precision, Recall, and F_1 indices, the results of MOGA-DBSCAN are better than SOGA-DBSCAN. The simultaneous use of both internal indices produced better results in all datasets, whereas MOGA-DBSCAN and SOGA-DBSCAN have the time complexities of $O(gn^2)$. MOGA-DBSCAN considers two objective functions and produces a diverse set of solutions. Therefore, there is a higher chance of producing the best solution.
(2) Based on the six validity indices, the results of MOGA-DBSCAN using Silhouette and Outlier-index as the objective functions are better than the results of MOGA-DBSCAN using Dunn and Silhouette as the objective functions. By using the Outlier-index, the false recognition of boundary points as the outlier is reduced, and more diversity in solutions is provided.

(3) The results of MOGA-DBSCAN using Silhouette and Outlier-index as the objective functions are similar or very close to the best clustering results of the DBSCAN algorithm for the six indices. However, MOGA-DBSCAN using Silhouette and Outlier-index as the objective function is fully automatic and requires no prior knowledge of the data.

Table 2. The *t*-test accepted hypothesis on the compound dataset.

	Algorithm	DBSCAN	SOGA-DBSCAN using Silhouette	SOGA-DBSCAN using Dunn	MOGA-DBSCAN using Silhouette and Dunn	MOGA-DBSCAN using Outlier-index and Silhouette
Jaccard index	DBSCAN	H_0	H_1	H_1	H_1	H_1
	SOGA-DBSCAN using Silhouette	H_1	H_0	H_0	H_1	H_1
	SOGA-DBSCAN using Dunn	H_1	H_0	H_0	H_1	H_1
	MOGA-DBSCAN using Silhouette and Dunn	H_1	H_1	H_1	H_0	H_1
	MOGA-DBSCAN using Outlier-index and Silhouette	H_1	H_1	H_1	H_1	H_0
Rand index	DBSCAN	H_0	H_1	H_1	H_1	H_1
	SOGA-DBSCAN using Silhouette	H_1	H_0	H_0	H_1	H_1
	SOGA-DBSCAN using Dunn	H_1	H_0	H_0	H_1	H_1
	MOGA-DBSCAN using Silhouette and Dunn	H_1	H_1	H_1	H_0	H_1
	MOGA-DBSCAN using Outlier-index and Silhouette	H_1	H_1	H_1	H_1	H_0
Min-kowski index	DBSCAN	H_0	H_1	H_1	H_1	H_1
	SOGA-DBSCAN using Silhouette	H_1	H_0	H_0	H_1	H_1
	SOGA-DBSCAN using Dunn	H_1	H_0	H_0	H_1	H_1
	MOGA-DBSCAN using Silhouette and Dunn	H_1	H_1	H_1	H_0	H_1
	MOGA-DBSCAN using Outlier-index and Silhouette	H_1	H_1	H_1	H_1	H_0
P Index	DBSCAN	H_0	H_1	H_1	H_1	H_1
	SOGA-DBSCAN using Silhouette	H_1	H_0	H_0	H_1	H_1
	SOGA-DBSCAN using Dunn	H_1	H_0	H_0	H_1	H_1
	MOGA-DBSCAN using Silhouette and Dunn	H_1	H_1	H_1	H_0	H_1

	MOGA-DBSCAN using Outlier-index and Silhouette	H_1	H_1	H_1	H_1	H_0
R Index	DBSCAN	H_0	H_1	H_1	H_1	H_1
	SOGA-DBSCAN using Silhouette	H_1	H_0	H_0	H_1	H_1
	SOGA-DBSCAN using Dunn	H_1	H_0	H_0	H_1	H_1
	MOGA-DBSCAN using Silhouette and Dunn	H_1	H_1	H_1	H_0	H_1
	MOGA-DBSCAN using Outlier-index and Silhouette	H_1	H_1	H_1	H_1	H_0

Table 3. The average of Jaccard, Rand, Minkowski, Precision, Recall, and F_1 indices in 30 runs on the compound dataset.

Algorithm / Index	DBSCAN	SOGA-DBSCAN using Silhouette	SOGA-DBSCAN using Dunn	MOGA-DBSCAN using Silhouette and Dunn	MOGA-DBSCAN using Outlier-index and Silhouette
Jaccard	0.9615	0.5578	0.4966	0.6953	0.9098
Rand	0.9903	0.7552	0.6964	0.8985	0.9761
Minkowski	0.1981	0.9253	1.0326	0.6373	0.3014
Precision	0.9825	0.6343	0.5863	0.7703	0.9485
Recall	0.9781	0.5983	0.5605	0.7383	0.9273
F_1	0.9802	0.6157	0.5731	0.7539	0.9377

Table 4. The *t*-test accepted hypothesis on the spiral dataset.

	Algorithm	DBSCAN	SOGA-DBSCAN using Silhouette	SOGA-DBSCAN using Dunn	MOGA-DBSCAN using Silhouette and Dunn	MOGA-DBSCAN using Outlier-index and Silhouette
Jaccard index	DBSCAN	H_0	H_1	H_1	H_1	H_0
	SOGA-DBSCAN using Silhouette	H_1	H_0	H_1	H_0	H_1
	SOGA-DBSCAN using Dunn	H_1	H_1	H_0	H_1	H_1
	MOGA-DBSCAN using Silhouette and Dunn	H_1	H_0	H_1	H_0	H_1
	MOGA-DBSCAN using Outlier-index and Silhouette	H_0	H_1	H_1	H_1	H_0
Rand index	DBSCAN	H_0	H_1	H_1	H_1	H_0
	SOGA-DBSCAN using Silhouette	H_1	H_0	H_1	H_0	H_1
	SOGA-DBSCAN using Dunn	H_1	H_1	H_0	H_1	H_1

	MOGA-DBSCAN using Silhouette and Dunn	H_1	H_0	H_1	H_0	H_1
	MOGA-DBSCAN using Outlier-index and Silhouette	H_0	H_1	H_1	H_1	H_0
Min-kowski index	DBSCAN	H_0	H_1	H_1	H_1	H_0
	SOGA-DBSCAN using Silhouette	H_1	H_0	H_1	H_0	H_1
	SOGA-DBSCAN using Dunn	H_1	H_1	H_0	H_1	H_1
	MOGA-DBSCAN using Silhouette and Dunn	H_1	H_0	H_1	H_0	H_1
	MOGA-DBSCAN using Outlier-index and Silhouette	H_0	H_1	H_1	H_1	H_0
P Index	DBSCAN	H_0	H_1	H_1	H_1	H_0
	SOGA-DBSCAN using Silhouette	H_1	H_0	H_1	H_0	H_1
	SOGA-DBSCAN using Dunn	H_1	H_1	H_0	H_1	H_1
	MOGA-DBSCAN using Silhouette and Dunn	H_1	H_0	H_1	H_0	H_1
	MOGA-DBSCAN using Outlier-index and Silhouette	H_0	H_1	H_1	H_1	H_0
R Index	DBSCAN	H_0	H_1	H_1	H_1	H_0
	SOGA-DBSCAN using Silhouette	H_1	H_0	H_1	H_0	H_1
	SOGA-DBSCAN using Dunn	H_1	H_1	H_0	H_1	H_1
	MOGA-DBSCAN using Silhouette and Dunn	H_1	H_0	H_1	H_0	H_1
	MOGA-DBSCAN using Outlier-index and Silhouette	H_0	H_1	H_1	H_1	H_0

Table 5. The average of Jaccard, Rand, Minkowski, Precision, Recall, and F_1 indices in 30 runs on the spiral dataset.

Algorithm / Index	DBSCAN	SOGA-DBSCAN using Silhouette	SOGA-DBSCAN using Dunn	MOGA-DBSCAN using Silhouette and Dunn	MOGA-DBSCAN using Outlier-index and Silhouette
Jaccard	1	0.6078	0.3152	0.6584	1
Rand	1	0.7884	0.3934	0.8287	1
Minkowski	0	0.7708	1.3530	0.6717	0
Precision	1	0.7047	0.3396	0.7521	1
Recall	1	0.6684	0.3608	0.7065	1
F_1	1	0.6861	0.3499	0.7286	1

Table 6. The *t*-test accepted hypothesis on the flame dataset.

	Algorithm	DBSCAN	SOGA-DBSCAN using Silhouette	SOGA-DBSCAN using Dunn	MOGA-DBSCAN using Silhouette and Dunn	MOGA-DBSCAN using Outlier-index and Silhouette
Jaccard index	DBSCAN	H_0	H_1	H_1	H_1	H_1
	SOGA-DBSCAN using Silhouette	H_1	H_0	H_1	H_1	H_1
	SOGA-DBSCAN using Dunn	H_1	H_1	H_0	H_1	H_1
	MOGA-DBSCAN using Silhouette and Dunn	H_1	H_1	H_1	H_0	H_1
	MOGA-DBSCAN using Outlier-index and Silhouette	H_1	H_1	H_1	H_1	H_0
Rand index	DBSCAN	H_0	H_1	H_1	H_1	H_1
	SOGA-DBSCAN using Silhouette	H_1	H_0	H_1	H_1	H_1
	SOGA-DBSCAN using Dunn	H_1	H_1	H_0	H_1	H_1
	MOGA-DBSCAN using Silhouette and Dunn	H_1	H_1	H_1	H_0	H_1
	MOGA-DBSCAN using Outlier-index and Silhouette	H_1	H_1	H_1	H_1	H_0
Min-kowski index	DBSCAN	H_0	H_1	H_1	H_1	H_1
	SOGA-DBSCAN using Silhouette	H_1	H_0	H_1	H_1	H_1
	SOGA-DBSCAN using Dunn	H_1	H_1	H_0	H_1	H_1
	MOGA-DBSCAN using Silhouette and Dunn	H_1	H_1	H_1	H_0	H_1
	MOGA-DBSCAN using Outlier-index and Silhouette	H_1	H_1	H_1	H_1	H_0
P Index	DBSCAN	H_0	H_1	H_1	H_1	H_1
	SOGA-DBSCAN using Silhouette	H_1	H_0	H_1	H_1	H_1
	SOGA-DBSCAN using Dunn	H_1	H_1	H_0	H_1	H_1
	MOGA-DBSCAN using Silhouette and Dunn	H_1	H_1	H_1	H_0	H_1
	MOGA-DBSCAN using Outlier-index and Silhouette	H_1	H_1	H_1	H_1	H_0
R Index	DBSCAN	H_0	H_1	H_1	H_1	H_1
	SOGA-DBSCAN using Silhouette	H_1	H_0	H_1	H_1	H_1
	SOGA-DBSCAN using Dunn	H_1	H_1	H_0	H_1	H_1

	MOGA-DBSCAN using Silhouette and Dunn	H_1	H_1	H_1	H_0	H_1
	MOGA-DBSCAN using Outlier-index and Silhouette	H_1	H_1	H_1	H_1	H_0

Table 7. The average of Jaccard, Rand, Minkowski, Precision, Recall, and F_1 indices in 30 runs on the flame dataset.

Algorithm / Index	DBSCAN	SOGA-DBSCAN using Silhouette	SOGA-DBSCAN using Dunn	MOGA-DBSCAN using Silhouette and Dunn	MOGA-DBSCAN using Outlier-index and Silhouette
Jaccard	0.9487	0.5241	0.5038	0.5322	0.9088
Rand	0.9721	0.5460	0.5390	0.5899	0.9511
Minkowski	0.2282	0.9204	0.9275	0.8747	0.3014
Precision	0.9684	0.5346	0.5216	0.5668	0.9415
Recall	0.9594	0.5297	0.5148	0.5412	0.9276
F_1	0.9638	0.5321	0.5181	0.5537	0.9344

Table 8. The *t*-test accepted hypothesis on the aggregation dataset.

	Algorithm	DBSCAN	SOGA-DBSCAN using Silhouette	SOGA-DBSCAN using Dunn	MOGA-DBSCAN using Silhouette and Dunn	MOGA-DBSCAN using Outlier-index and Silhouette
Jaccard index	DBSCAN	H_0	H_1	H_1	H_1	H_0
	SOGA-DBSCAN using Silhouette	H_1	H_0	H_1	H_1	H_1
	SOGA-DBSCAN using Dunn	H_1	H_1	H_0	H_1	H_1
	MOGA-DBSCAN using Silhouette and Dunn	H_1	H_1	H_1	H_0	H_1
	MOGA-DBSCAN using Outlier-index and Silhouette	H_0	H_1	H_1	H_1	H_0
Rand index	DBSCAN	H_0	H_1	H_1	H_1	H_0
	SOGA-DBSCAN using Silhouette	H_1	H_0	H_1	H_1	H_1
	SOGA-DBSCAN using Dunn	H_1	H_1	H_0	H_1	H_1
	MOGA-DBSCAN using Silhouette and Dunn	H_1	H_1	H_1	H_0	H_1

	MOGA-DBSCAN using Outlier-index and Silhouette	H_0	H_1	H_1	H_1	H_0
Min-kowski index	DBSCAN	H_0	H_1	H_1	H_1	H_0
	SOGA-DBSCAN using Silhouette	H_1	H_0	H_1	H_1	H_1
	SOGA-DBSCAN using Dunn	H_1	H_1	H_0	H_1	H_1
	MOGA-DBSCAN using Silhouette and Dunn	H_1	H_1	H_1	H_0	H_1
	MOGA-DBSCAN using Outlier-index and Silhouette	H_0	H_1	H_1	H_1	H_0
P Index	DBSCAN	H_0	H_1	H_1	H_1	H_0
	SOGA-DBSCAN using Silhouette	H_1	H_0	H_1	H_1	H_1
	SOGA-DBSCAN using Dunn	H_1	H_1	H_0	H_1	H_1
	MOGA-DBSCAN using Silhouette and Dunn	H_1	H_1	H_1	H_0	H_1
	MOGA-DBSCAN using Outlier-index and Silhouette	H_0	H_1	H_1	H_1	H_0
R Index	DBSCAN	H_0	H_1	H_1	H_1	H_0
	SOGA-DBSCAN using Silhouette	H_1	H_0	H_1	H_1	H_1
	SOGA-DBSCAN using Dunn	H_1	H_1	H_0	H_1	H_1
	MOGA-DBSCAN using Silhouette and Dunn	H_1	H_1	H_1	H_0	H_1
	MOGA-DBSCAN using Outlier-index and Silhouette	H_0	H_1	H_1	H_1	H_0

Table 9. The average of Jaccard, Rand, Minkowski, Precision, Recall, and F_1 indices in 30 runs on the aggregation dataset.

Algorithm / Index	DBSCAN	SOGA-DBSCAN using Silhouette	SOGA-DBSCAN using Dunn	MOGA-DBSCAN using Silhouette and Dunn	MOGA-DBSCAN using Outlier-index and Silhouette
Jaccard	0.9966	0.4699	0.6169	0.7486	0.9966
Rand	0.9993	0.5889	0.7594	0.9273	0.9993

Minkowski	0.0587	1.2371	0.8987	0.5794	0.0587
Precision	0.9987	0.5314	0.6889	0.8541	0.9987
Recall	0.9971	0.4915	0.6501	0.8111	0.9971
F_1	0.9978	0.5106	0.6689	0.8321	0.9978

Table 10. The *t*-test accepted hypothesis on the path-based dataset.

	Algorithm	DBSCAN	SOGA-DBSCAN using Silhouette	SOGA-DBSCAN using Dunn	MOGA-DBSCAN using Silhouette and Dunn	MOGA-DBSCAN using Outlier-index and Silhouette
Jaccard index	DBSCAN	H_0	H_1	H_1	H_1	H_1
	SOGA-DBSCAN using Silhouette	H_1	H_0	H_0	H_1	H_1
	SOGA-DBSCAN using Dunn	H_1	H_0	H_0	H_1	H_1
	MOGA-DBSCAN using Silhouette and Dunn	H_1	H_1	H_1	H_0	H_1
	MOGA-DBSCAN using Outlier-index and Silhouette	H_1	H_1	H_1	H_1	H_0
Rand index	DBSCAN	H_0	H_1	H_1	H_1	H_1
	SOGA-DBSCAN using Silhouette	H_1	H_0	H_0	H_1	H_1
	SOGA-DBSCAN using Dunn	H_1	H_0	H_0	H_1	H_1
	MOGA-DBSCAN using Silhouette and Dunn	H_1	H_1	H_1	H_0	H_1
	MOGA-DBSCAN using Outlier-index and Silhouette	H_1	H_1	H_1	H_1	H_0
Minkowski index	DBSCAN	H_0	H_1	H_1	H_1	H_1
	SOGA-DBSCAN using Silhouette	H_1	H_0	H_0	H_1	H_1
	SOGA-DBSCAN using Dunn	H_1	H_0	H_0	H_1	H_1
	MOGA-DBSCAN using Silhouette and Dunn	H_1	H_1	H_1	H_0	H_1
	MOGA-DBSCAN using Outlier-index and Silhouette	H_1	H_1	H_1	H_1	H_0

P Index	DBSCAN	H_0	H_1	H_1	H_1	H_1
	SOGA-DBSCAN using Silhouette	H_1	H_0	H_0	H_1	H_1
	SOGA-DBSCAN using Dunn	H_1	H_0	H_0	H_1	H_1
	MOGA-DBSCAN using Silhouette and Dunn	H_1	H_1	H_1	H_0	H_1
	MOGA-DBSCAN using Outlier-index and Silhouette	H_1	H_1	H_1	H_1	H_0
R Index	DBSCAN	H_0	H_1	H_1	H_1	H_1
	SOGA-DBSCAN using Silhouette	H_1	H_0	H_0	H_1	H_1
	SOGA-DBSCAN using Dunn	H_1	H_0	H_0	H_1	H_1
	MOGA-DBSCAN using Silhouette and Dunn	H_1	H_1	H_1	H_0	H_1
	MOGA-DBSCAN using Outlier-index and Silhouette	H_1	H_1	H_1	H_1	H_0

Table 11. The average of Jaccard, Rand, Minkowski, Precision, Recall, and F_1 indices in 30 Runs on the path-based dataset.

Algorithm / Index	DBSCAN	SOGA-DBSCAN using Silhouette	SOGA-DBSCAN using Dunn	MOGA-DBSCAN using Silhouette and Dunn	MOGA-DBSCAN using Outlier-index and Silhouette
Jaccard	0.8982	0.5259	0.5253	0.6425	0.7929
Rand	0.9642	0.6374	0.6310	0.8624	0.9221
Minkowski	0.3279	0.7083	0.7962	0.6429	0.4836
Precision	0.9362	0.5715	0.5693	0.7584	0.8657
Recall	0.9092	0.5427	0.5401	0.7178	0.8293
F_1	0.9225	0.5567	0.5543	0.7375	0.8471

5. CONCLUSIONS

In this study, a hybrid algorithm using MOGA has been proposed to determine the parameters of the DBSCAN algorithm automatically. In this algorithm, a set of values for *Eps* and *MinPts* parameters is generated by considering the internal indices as the objective functions. Therefore, the most suitable solution could be selected depending on the application. To providing more diversity and quality in produced solutions, a new internal index named Outlier-index is proposed. Furthermore, the Delaunay triangulation

technique is used for determining the initial bounds of the *Eps* and *MinPts* parameters to reduce the number of generations. This technique needs no input parameters. Hence, MOGA-DBSCAN is a fully automatic technique that needs no input parameters.

The DBSCAN algorithm is a heuristic technique with the time complexity of $O(n\log n)$ that is more effective than the proposed meta-heuristic technique with the time complexity of $O(gn^2)$ with respect to computations. Nevertheless, DBSCAN and its improved versions need to determine at least one parameter and interact with users. Moreover, it is not suitable for applications where there is no prior knowledge of the data and online applications.

In this study, the results of evaluating MOGA-DBSCAN on a number of standard datasets are compared with the best results obtained from DBSCAN and those of SOGA-DBSCAN using Dunn or Silhouette as the objective functions. As denoted by the results in the previous section, the results of MOGA-DBSCAN, which determines the parameters automatically, are close to the best results of DBSCAN clustering. It should be noted that, the results obtained by using the OPTICS algorithm for determining the parameters of the DBSCAN algorithm are found to be superior using the trial and error test on the various *MinPts* values for each dataset. In contrast, the average of the best solutions for 30 runs is taken into consideration in MOGA-DBSCAN, and the parameters are determined automatically.

Based on the accepted hypothesis in the *t*-test for the comparison between the results of each pair of algorithms that is conducted on five datasets, the clustering results of MOGA-DBSCAN are found to be better than those of SOGA-DBSCAN employing Dunn or Silhouette as the objective functions for the Jaccard, Rand, Minkowski, Precision, Recall, and F_1 indices. As a result, the simultaneous use of both internal indices produced better results in all datasets, whereas MOGA-DBSCAN and SOGA-DBSCAN have the same time complexity.

For the future works, MOGA-DBSCAN could be developed to clustering moving objects. In the moving objects clustering, the density of the clusters changed at each time interval, and the parameters of DBSCAN need to be re-determined dynamically.

REFERENCES

1. J. Han, J. Pei, and M. Kamber, *Data Mining: Concepts and Techniques*, Elsevier, 2011.
2. C. Yang, F. Wang, and B. Huang, "Internet traffic classification using dbscan," in *Proceedings of IEEE WASE International Conference on Information Engineering*, Vol. 2, 2009, pp. 163-166.
3. Z. Francis, C. Villagrasa, and I. Clairand, "Simulation of DNA damage clustering after proton irradiation using an adapted DBSCAN algorithm," *Computer Methods and Programs in Biomedicine*, Vol. 101, 2011, pp. 265-270.
4. K. M. Mahmoud, M. A. Ismail, and N. M. Ghanem, "Vscan: an enhanced video summarization using density-based spatial clustering," in *Proceedings of International Conference on Image Analysis and Processing*, 2013, pp. 733-742.
5. Y. M. ElBarawy, R. F. Mohamed, and N. I. Ghali, "Improving social network community detection using DBSCAN algorithm," in *Proceedings of IEEE World Symposium on Computer Applications and Research*, 2014, pp. 1-6.

6. Z. Gui, H. Yu, and Y. Tang, "Locating traffic hot routes from massive taxi tracks in clusters," *Journal of Information Science and Engineering*, Vol. 32, 2016, pp. 113-131.
7. J. Shen, X. Hao, Z. Liang, Y. Liu, W. Wang, and L. Shao, "Real-time super pixel segmentation by DBSCAN clustering algorithm," *IEEE Transactions on Image Processing*, Vol. 25, 2016, pp. 5933-5942.
8. M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: ordering points to identify the clustering structure," in *ACM Sigmod Record*, Vol. 28, 1999, pp. 49-60.
9. L. Duan, L. Xu, F. Guo, J. Lee, and B. Yan, "A local-density based spatial clustering algorithm with noise," *Information Systems*, Vol. 32, 2007, pp. 978-986.
10. P. Liu, D. Zhou, and N. Wu, "VDBSCAN: varied density based spatial clustering of applications with noise," in *Proceedings of IEEE International Conference on Service Systems and Service Management*, 2007, pp. 1-4.
11. S. Mitra and J. Nandy, "Kddclus: A simple method for multi-density clustering," in *Proceedings of International Workshop on Soft Computing Applications and Knowledge Discovery*, 2011, pp. 72-77.
12. H. Darong and W. Peng, "Grid-based DBSCAN algorithm with referential parameters," *Physics Procedia*, Vol. 24, 2012, pp. 1166-1170.
13. A. Smiti and Z. Elouedi, "DBSCAN-GM: An improved clustering method based on Gaussian Means and DBSCAN techniques," in *Proceedings of IEEE 16th International Conference on Intelligent Engineering Systems*, 2012, pp. 573-578.
14. K. Diao, Y. Liang, and J. Fan, "An improved DBSCAN algorithm using local parameters," in *Proceedings of International Conference on Artificial Intelligence*, 2018, pp. 3-12.
15. L. M. Abualigah, A. T. Khader, and E. S. Hanandeh, "A new feature selection method to improve the document clustering using particle swarm optimization algorithm," *Journal of Computational Science*, Vol. 25, 2018, pp. 456-466.
16. A. N. Benaichouche, H. Oulhadj, and P. Siarry, "Improved spatial fuzzy c-means clustering for image segmentation using PSO initialization, Mahalanobis distance and post-segmentation correction," *Digital Signal Processing*, Vol. 23, 2013, pp. 1390-1400.
17. J. Chen, "Hybrid clustering algorithm based on pso with the multidimensional asynchronism and stochastic disturbance method," *Journal of Theoretical and Applied Information Technology*, Vol. 46, 2012, pp. 434-440.
18. E. Hancer, C. Ozturk, and D. Karaboga, "Artificial bee colony based image clustering method," in *Proceedings of IEEE Congress on Evolutionary Computation*, 2012, pp. 1-5.
19. A. Karami and M. Guerrero-Zapata, "A fuzzy anomaly detection system based on hybrid pso-kmeans algorithm in content-centric networks," *Neurocomputing*, Vol. 149, 2015, pp. 1253-1269.
20. R. Kuo and F. E. Zulvia, "Automatic clustering using an improved artificial bee colony optimization for customer segmentation," *Knowledge and Information Systems*, Vol. 57, 2018, pp. 331-357.
21. Y. Li, J. Chen, R. Liu, and J. Wu, "A spectral clustering-based adaptive hybrid multiobjective harmony search algorithm for community detection," in *Proceedings of*

IEEE Congress on Evolutionary Computation, 2012, pp. 1-8.
22. G. E. Tsekouras, "A simple and effective algorithm for implementing particle swarm optimization in RBF network's design using input-output fuzzy clustering," *Neurocomputing*, Vol. 108, 2013, pp. 36-44.
23. A. Karami and R. Johansson, "Choosing dbscan parameters automatically using differential evolution," *International Journal of Computer Applications*, Vol. 91, 2014, pp. 1-11.
24. B. Issac and N. Israr, *Case Studies in Intelligent Computing: Achievements and Trends*, CRC Press, UK, 2014.
25. U. Maulik, S. Bandyopadhyay, and A. Mukhopadhyay, *Multiobjective Genetic Algorithms for Clustering: Applications in Data Mining and Bioinformatics*, Springer Science & Business Media, 2011.
26. M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, Vol. 96, 1996, pp. 226-231.
27. D. Kalyanmoy, *Multi Objective Optimization Using Evolutionary Algorithms*, John Wiley and Sons, NY, 2001.
28. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, Vol. 6, 2002, pp. 182-197.
29. B. Delaunay, "Sur la sphere vide," Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk, 1934, pp. 793-800.
30. P. Roy and J. Mandal, "A novel spatial fuzzy clustering using delaunay triangulation for large scale gis data (nsfcdt)," *Procedia Technology*, Vol. 6, 2012, pp. 452-459.
31. X. Yang and W. Cui, "A novel spatial clustering algorithm based on delaunay triangulation," *Software Engineering & Applications*, Vol. 3, 2010, pp. 141-149.
32. X. Zhong and M. Duckham, "Characterizing the shapes of noisy, non-uniform, and disconnected point clusters in the plane," *Computers, Environment and Urban Systems*, Vol. 57, 2016, pp. 48-58.
33. P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, Vol. 20, 1987, pp. 53-65.
34. J. C. Dunn, "Well-separated clusters and optimal fuzzy partitions," *Journal of Cybernetics*, Vol. 4, 1974, pp. 95-104.
35. C. D. Manning, P. Raghavan, and H. Schutze, *Introduction to Information Retrieval*, Cambridge University Press, UK, 2008.
36. R. L. Ott and M. T. Longnecker, *An Introduction to Statistical Methods and Data Analysis*, Nelson Education, Canada, 2015.
37. K. M. Ramachandran and C. P. Tsokos, *Mathematical Statistics with Applications in R*, Elsevier, USA, 2014.
38. D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in *Foundations of Genetic Algorithms*, Vol. 1, Elsevier, 1991, pp. 69-93.
39. Clustering Datasets, http://cs.joensuu.fi/sipu/datasets.

Zeinab Falahiazar received the B.S. degree in Computer Engineering (Software) from Shariaty Technical College, in 2007, and received the M.S. degree in Computer Engineering (Software) from Islamic Azad University, Tehran South Branch, in 2010. She is currently a Ph.D. student in the Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran. Her main research interests include spatial data mining.

Alireza Bagheri received his B.S. and M.S. degrees in Computer Engineering from Sharif University of Technology (SUT) at Tehran, in 1996 and 1998 respectively. He received his Ph.D. degree in Computer Science from Amirkabir University of Technology (AUT) at Tehran, in 2005. Currently he is an Associate Professor in the Computer Engineering and IT Department at Amirkabir University of Technology (AUT) at Tehran. His research interests include computational geometry, graph algorithms, bigdata analysis, and social network analysis.

Midia Reshadi is currently an Assistant Professor in Computer Engineering Department at Science and Research Branch of Azad University since 2010. His research interest is network-on-chip including performance and cost improvement in topology, routing and application-mapping design levels of various types of NoCs such as 3D, photonic and wireless. Recently, he has started carrying out research in NoC based deep neural network accelerators and silicon interposer based NoC with his team consists of MSc and Ph.D. students.