Journal of Artificial Intelligence and Data Mining (JAIDM), Vol. 10, No. 3, 2022, 321-332.

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Research paper

Determining parameters of DBSCAN Algorithm in Dynamic Environments Automatically using Dynamic Multi-objective Genetic Algorithm

Zeinab Falahiazar[1], Alireza Bagheri[2*] and Midia Reshadi[1]

1. Department of Computer Engineering, Science and Research branch, Islamic Azad University,Tehran, Iran.
2. Department of Computer Engineering Amirkabir University of Technology,Tehran, Iran.

Article Info

Article History:
Received 20 December 2021
Revised 26 April 2022
Accepted 31 May 2022

DOI:10.22044/JADM.2022.11499.2309

Keywords:

Density-based Clustering, DBSCAN, Dynamic Multi-objective Optimization, Clustering Moving Objects, Cluster Validity Index.

**Corresponding author: ar_bagheri@aut.ac.ir(A. Bagheri).*

Abstract

The spatio-temporal (ST) clustering is a relatively new field in data mining with a great popularity, especially in the geographic information. The moving objects are a type of ST data where the available information on these objects includes their last position. The strategy of performing the clustering operation on all-time sequences is used for clustering the moving objects. The problem with density-based clustering, which uses this strategy, is that the density of clusters may change at any point in time due to the displacement of points. Hence, the input parameters of an algorithm like DBSCAN used to cluster the moving objects will change, and have to be determined again. The DBSCAN-based methods have been proposed so far, assuming that the value of the input parameters is fixed over time and does not provide a solution for their automatic determination. Nonetheless, with the objects moving and the density of the clusters changing, these parameters have to be determined appropriately again at each time interval. This work uses a dynamic multi-objective genetic algorithm in order to determine the parameters of the DBSCAN algorithm dynamically and automatically to solve this problem. The proposed algorithm in each time interval uses the clustering information of the previous time interval to determine the parameters. The Beijing traffic control data is used as a moving dataset in order to evaluate the proposed algorithm. The experiments show that using the proposed algorithm for dynamic determination of the DBSCAN input parameters outperforms DBSCAN with fixed input parameters over time in terms of the Silhouette and Outlier indices.

1. Introduction

The spatio-temporal (ST) clustering refers to the process of grouping objects based on their spatial and temporal similarities. Indeed, ST data clustering does not much differ from spatial data clustering. All the related studies try to introduce the concept of time as a threshold into data or algorithms either by distance functions or even to transform the problem of ST clustering into a multi-stage spatial data clustering. The strategy of performing clustering operations on all-time sequences is used for clustering the moving objects [1]. The problem with density-based clustering that uses this strategy is that the density of clusters may change at any point in time due to the displacement of points (Figure 1). Hence, the input parameters of these algorithms change and need to be reset. The DBSCAN algorithm is a widely used algorithm among the density-based clustering algorithms. The algorithm needs two input parameters, Eps and MinPts, and can detect clusters with various shapes and outliers. However, determining the input parameters of this algorithm is hard, and the value of the parameters has a significant effect on the clustering result. An automated method called MOGA-DBSCAN has been presented in [2] in order to determine these parameters in a static environment. In this method, the DBSCAN clustering is considered as a multi-objective optimization problem. To this end, a multi-objective genetic algorithm is used, with each chromosome consisting of two genes representing the values of the parameters Eps and MinPts. Nonetheless, no method has been presented to determine the input parameters in dynamic environments, and in all solutions for clustering in dynamic environments, the input values of the parameters have been considered constant over time. For instance, performing the DBSCAN algorithm on all-time sequences is used in [3] to cluster the moving objects. In this method, an incremental technique is used to reduce the computational cost. In this method, first, the DBSCAN algorithm is performed, and the position of the points is stored in the memory. In the following time intervals, the points are checked, and only those that have moved are placed in one of the clusters according to their distance from the clusters. Nevertheless, this method considers the DBSCAN parameters constant over time, and does not provide a solution for their automatic determination. However, these parameters have to be determined again appropriately in each period by moving the points and changing the density of the clusters.

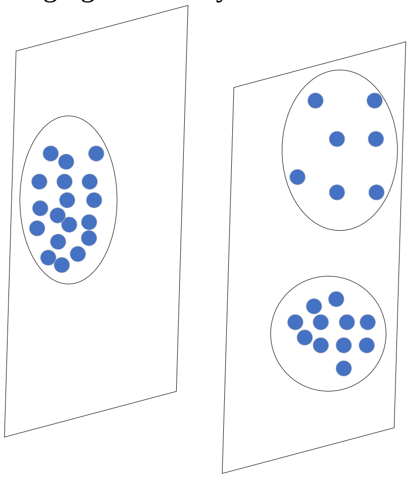


Figure 1. Changing density of clusters related to moving objects.

Among the clustering methods related to the clustering of moving objects, trajectory clustering and incremental clustering can be cited. Trajectory clustering refers to the clustering of moving objects that behave similarly over time (similar paths or similar destinations) and differs from the clustering of moving objects that occurs at any given time interval. In other words, in trajectory clustering, a cluster is a fixed set of objects over their lifetime, whereas in the moving object clustering, the contents of a cluster may change over time. The incremental clustering algorithms are used to cluster the moving objects [3, 4]. In these algorithms, in order to reduce the computational cost and increase the speed, the clustering algorithm is usually fully performed in the first time interval, and in the subsequent time intervals, the same initial clusters are used to cluster the moving objects so that each data object is placed inside one of the clusters according to its new location; these algorithms are practical only when a small percentage of objects are moved, and this is not true of the problem of moving objects where all points can potentially move. In the following, we review a number of proposed solutions that use incremental clustering. In [5], a DBSCAN-based incremental clustering method is proposed. In this method, first, the DBSCAN algorithm is performed, and in the next time intervals, the dataset is checked incrementally so that each point is located in one of the clusters according to the DBSCAN algorithm. The paper has assumed that removing or adding a new object only affects data objects that are in its vicinity. Updating is done in batches, and it is assumed that these updated items are a small percentage of the dataset. Hence, the density of the clusters is not affected, and there is no need to determine the DBSCAN parameters again. However, this assumption is not true for moving objects. In [6], an incremental clustering method based on DBSCAN provided that instead of adding new data to the clusters each time, some new data are first clustered, and the new clusters are merged with the previous clusters. This is used where a large number of new data is entered at any one time. This method, like the previous method, assumes that the data movement consists of only a small percentage of data, and the density of the clusters does not change much. In a sharp change in density, the DBSCAN algorithm must be performed again. However, it does not offer a solution to determine the parameters of the DBSCAN algorithm. In [7, 8], the authors have used the search space segmentation to place the new object in the appropriate cluster. For doing so, the search space is divided into k parts, the DBSCAN clustering algorithm is performed in each part, the dense regions are merged, and a new point is added to the nearest part cluster. This method increases the clustering speed. In this method, too, it is assumed that the DBSCAN parameters do not change over time. In [9], first, the EPS parameter values are determined, and multi-density clustering is performed. The new points then join the created clusters. In the incremental methods, it is assumed that the DBSCAN parameters do not change over time, and it is necessary to re-implement the DBSCAN and reset its parameters if there is a large change in the structure of the clusters. Reference [10] presents a parallel method for the incremental data flow clustering according to the DBSCAN algorithm. The segmentation technique is used in this method, and each part is incrementally clustered. This method relies on the parameters of the DBSCAN algorithm, determined by the user at the beginning of the clustering, and is assumed not to change until the end of the clustering.

The solutions proposed for clustering the moving objects based on the DBSCAN algorithm presented so far do not provide a solution for the automatic and dynamic determination of the input parameters. In this paper, a method for automatic determination of the DBSCAN algorithm parameters in dynamic environments is presented so that it can be used in conjunction with moving object clustering methods when there are significant changes in the environment. The dynamic multi-objective genetic algorithm is used to determine the parameters of the DBSCAN algorithm dynamically and automatically to solve this problem in the proposed solution. The dynamic multi-objective optimization algorithms have the opportunity to find optimal solutions at that time before changes occur (τ_T). After the changes in the problem, the values of the objective functions of the solutions change and the algorithm once again has the opportunity to find the optimal solutions before the changes occur. The process goes on up to the last generation of the algorithm. The proposed method will use the information of the previous clustering to determine the parameters, which reduces the calculations such as redefining the parameter range. In the proposed solution, the clustering problem is considered as a dynamic multi-objective optimization problem. By using this algorithm, new responses are generated in each period based on the changes in the environment. The advantage of using a multi-objective optimization algorithm is to generate a set of responses instead of a single one that enables the user to select the proper clustering for the dataset about which there is no prior knowledge. Simultaneous optimization of more than one index increases the quality of clustering outcomes too.

The rest of the paper is organized as what follows. Section 2 describes the related concepts such as the DBSCAN algorithm, ST data, multi-objective optimization, Delaunay Triangulation, and the MOGA-DBSCAN algorithm. Section 3 shows the proposed algorithm for determining the DBSCAN parameters in dynamic environments. The implementation results are evaluated in Section 4, and 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 in dynamic environments.

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 [11]. 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:

- Core point: a point that has at least the minimum number of points (MinPts) in the neighborhood radius (Eps).
- 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.
- 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_l = q and p_n = p such that p_{i+1} is directly density-reachable from pi.

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.

The 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(nlogn). If the Eps and MinPts parameters are adjusted appropriately, the algorithm will effectively determine the clusters of arbitrary shapes.

2.2. ST data

ST clustering is a relatively new field in data mining with a great popularity, especially in geographic information, because of the prevalence of all types of location-based devices that instantly record the location, time or environmental characteristics of an object. In real applications, several different types of ST data exist. In [1], a possible classification of data objects is shown based on two temporal and spatial dimensions. In the following, we briefly describe the main classes of data types obtained for the data objects.

- ST events: A basic sample of ST information is ST events like earthquakes gained by sensors or geographically referenced records of an epidemic.
- Geo-referenced variables: When it is possible to see the evolution of a phenomenon over time in a fixed location.
- Geo-referenced time series: In a more complex situation, it is possible to store the whole history of an evolving object. Hence, it is possible to provide time series (with geo-reference) for the measured variables.
- Moving objects: When the spatial location of a data object changes over time, we face the moving objects. In its simplest form, the information available about these objects includes their last position, and does not maintain a sequence of previous positions like instantaneous monitoring of vehicles for security uses. Clustering is applied to any time sequence in the problem of clustering moving objects. The clustering algorithms are commonly used to cluster the actual data such as a group of migrating animals and the number of cars moving in a city [3].
- Trajectories: When the entire history of a moving object is stored and available for analysis. Trajectories explain the mobility behavior of objects, and so clustering can be used to identify the objects that behave similarly.

2.3. Multi-objective optimization

Multi-objective 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 [12, 13]:

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

A solution $\bar{x}$ is a vector of n decision variables, which satisfies the m inequality ($g_i(\bar{x})$) and the p equality constraints ($h_i(\bar{x})$) and optimizes the vector function ($\bar{f}(\bar{x})$) 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 $\overline{x^*}$ can be regarded as a part of the Pareto optimal front if and only if there is no vector $\bar{x}$ that can dominate $\overline{x^*}$:

$$\begin{aligned}
&\forall i \in 1, 2, \ldots, k, f_i(\bar{x}) \geq f_i(\overline{x^*}) \\
&\exists i \in 1, 2, \ldots, k, f_i(\bar{x}) > f_i(\overline{x^*})
\end{aligned} \qquad (2)$$

where k is the number of objective functions.

The optimization problems are divided into static and dynamic classes [14]. The objective functions do not change over time in static optimization problems, yet the objective functions change over time in the dynamic optimization problems. Unlike the static multi-objective optimization problems, the optimal solutions change over time in dynamic multi-objective optimization problems, and the dynamic multi-objective optimization algorithm must adapt to these changes. In better words, dynamic optimization tries to follow the optimal solutions. The dynamic multi-objective optimization algorithms could find the optimal solutions before the changes happen, and the values of the solutions' objective functions change after changes in the problem, and the algorithm once again could find the optimal solutions before changes happen. This goes on until the last generation of the algorithm. The parameters of dynamic multi-objective optimization algorithms are used to adjust their performance. These parameters are defined as follows:

- τ: This parameter indicates the generation number of the current population.
- τ_T: This parameter is called "change frequency" and the environment changes once for each τ_T generation. In other words, the dynamic multi-objective optimization algorithm could find optimal solutions up to τ_T generation, and then the environment changes, and the dynamic multi-objective optimization algorithm has the opportunity to find optimal solutions again up to the τ_T generation. For instance, if $\tau 1$ is the first generation and τ_T is ten, then the environment changes in the eleventh generation, the dynamic multi-objective optimization algorithm has the chance to search for optimal solutions up to the eleventh generation.
- τ_{max}: This is the maximum number of generations that a dynamic multi-objective optimization algorithm generates. In other words, this parameter determines the end condition of the dynamic multi-objective optimization algorithm.

Non-Sorting Genetic Algorithm–II (NSGA-II algorithm) [15], one of the most successful multi-objective optimization algorithms and a reference algorithm for multi-objective optimization researchers, was developed by Deb *et al.* for the dynamic multi-objective optimization problems. In a dynamic state, this algorithm re-evaluates the entire population by detecting the occurrence of a change in the environment, and one of the following two approaches is adopted to increase the diversity:

- The percentage of the population is replaced by random solutions (Dynamic NSGA–II–A).
- Some members of the population are randomly selected and mutated (Dynamic NSGA–II–B).

2.4 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) [16, 17]. This triangulation developed by Boris Delaunay and is used in different applications of geographic information systems (GIS).

$$\left\{ x \in R^2 \mid \forall_{j \neq i}, d\left(x, p_i\right) \leq d\left(x, p_j\right) \right\} \qquad (3)$$

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. In Equation 3, d is the distance function. The Voronoi diagram is formed by n Voronoi 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 pi and p_j are connected by an edge if the borders of their Voronoi regions have a shared line. Figure 2 shows the Delaunay triangulation of 15 points.

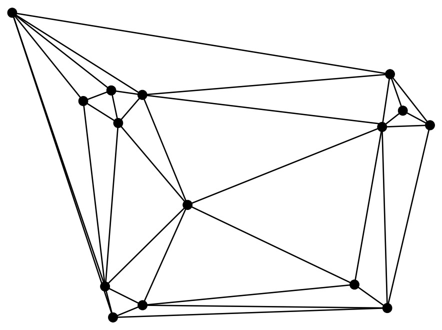


Figure 2. Delaunay triangulation of 15 points [17].

2.5. MOGA-DBSCAN

Algorithm 1 shows the steps in the MOGA-DBSCAN [2]. In step 1, the initial population is randomly determined in the bounds specified by the Delaunay triangulation algorithm. The Eps and MinPts bounds are determined as follows 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.

In step 2, the DBSCAN algorithm is first run by each member of the initial population, which are the parameters of the DBSCAN algorithm, and the clustering results are evaluated by internal validation indices used as the objective functions, and the fitting values are obtained for the initial population. In step 5, the mutation and cross-over 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 6). The different values of the DBSCAN parameters and the validity indices form the search space.

Algorithm 1: Pseudo-code of MOGA- DBSCAN [2]

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

At the end of the MOGA run time, the final set of the near-Pareto-optimal solutions includes a number of non-dominated solutions (step 7). There is no interaction with the user in this algorithm to determine the solutions. A small set of the best solutions according to the indices used as the objective function is provided at the end of the algorithm. 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 [18, 19]. Also in [2], a new index based on the outliers detected by a clustering algorithm is presented as follows:

$$os_i = min_{1\le j\le m}\left(d_{ij}\right)$$
$$Outlier-index = \frac{\sum os_i}{n} \qquad (4)$$

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. In the proposed algorithm, two internal cluster validity indices, Silhouette [20] and outlier, are used as the objective functions. The Silhouette index is defined as follows:

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 (s_i) can be defined as follows:

$$s_i = \frac{b_i - a_i}{\max\left\{a_i, b_i\right\}} \qquad (5)$$

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

$$S = \frac{1}{n}\sum_{i=1}^{n} s_i \qquad (6)$$

where n is the number of all the data points. The value of the Silhouette index changes between -1 and 1, and the higher values indicate better clustering results.

3. Proposed Algorithm for Determining DBSCAN Parameters in Dynamic Environments

The solutions proposed for clustering moving objects based on the DBSCAN algorithm suggested do not provide a solution for automatic and dynamic determination of input parameters. A dynamic multi-objective genetic algorithm (DMOGA) is used to determine the parameters of the DBSCAN algorithm dynamically and automatically to solve this problem in the proposed solution. Figure 3 shows the flowchart

of the proposed algorithm. At the beginning of each period, it is necessary to determine the Eps and MinPts bounds in DMOGA appropriately to generate high-quality solutions. The Delaunay triangulation is employed to determine the Eps and MinPts bounds. In the next three steps, the MOGA algorithm is used to determine the appropriate value of the parameters. After τ_T generations, the current time period ends,
and the environment changes. At this time, the determined parameters can be used by the clustering algorithms. If the number of generations has not reached τ_{max}, the DMOGA algorithm goes back to the first step and tries to determine new values for the parameters due to changes in the environment.
Algorithm 2 shows the steps in the DMOGA-DBSCAN. The environment changes τ_{max}/τ_T times. In step 5, selection of the initial population is based on the type of dynamic bi-objective optimization algorithm using the previous clustering response set. In step 8, the dynamic bi-objective optimization algorithm could find the optimal solutions in that period before the changes happen (τ_T). After the changes in the problem, the values of the objective functions of the solutions change, and the algorithm once again has the opportunity to find the optimal solutions before the changes occur. This process goes on till the last generation of the algorithm (τ_{max}).

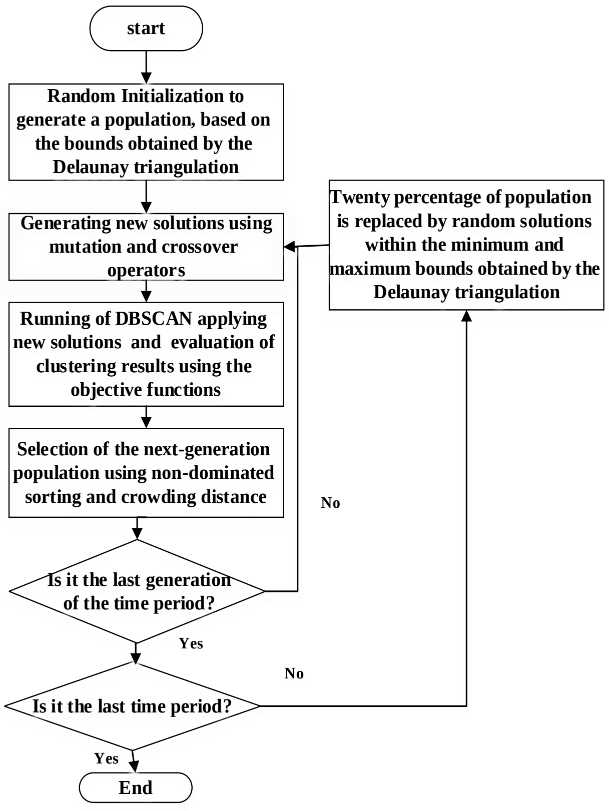

start
Random Initialization to generate a population, based on the bounds obtained by the Delaunay triangulation
Generating new solutions using mutation and crossover operators
Running of DBSCAN applying new solutions and evaluation of clustering results using the objective functions
Selection of the next-generation population using non-dominated sorting and crowding distance
Is it the last generation of the time period?
No
Yes
Is it the last time period?
No
Yes
Twenty percentage of population is replaced by random solutions within the minimum and maximum bounds obtained by the Delaunay triangulation
End

Figure 3. Flowchart of DMOGA-DBSCAN.

New responses are generated in each period based on the changes in the environment using this algorithm. The advantage of using a multi-objective optimization algorithm is to create a set of responses instead of a single one, allowing the user to select the appropriate clustering for the dataset about which there is no prior knowledge. The simultaneous optimization of more than one cluster validity index enhances the quality of clustering results.

Algorithm 2: Pseudo-code of DMOGA- DBSCAN

```
      // P and PF are outputs of Function //
   // P = Parent Population, Q = Offspring Population, PF = Pareto Front, S= Input Dataset //
      Function DMOGA-DBSCAN
1   For j=1 to $\tau_{max}/\tau_T$
2     If P is Empty
3       Initialize the P randomly within the minimum and maximum bounds;
4     Else
5       Twenty percentage of P is replaced by random solutions within the minimum and maximum bounds
6     End-if
7     Run objective function1 and objective function2 over the results of DBSCAN(S,P) );
8     For i=1 to $\tau_T$
9       Q = Union(Mutation(P), Crossover(P) );
10      Run objective function1 and objective function2 over the results of DBSCAN(S,Q) ;
11      [P PF] = Selection(Q,P);
12    End-For
13  End-For
14  End-Function
```

4. Implementation and Evaluation

In this work, NSGA-II is used as a non-dominated, sorting-based, multi-objective evolutionary algorithm to implement MOGA-DBSCAN, and DNSGA-II–A is used to determine the DBSCAN parameters (DMOGA-DBSCAN) dynamically. Also two internal cluster validity indices, Silhouette and Outlier, are used as the objective functions. The rates of mutation and cross-over operators and population size in this paper are, respectively, 0.5, 0.9, and 30, used in all implementations. A synthetic dataset is generated by 399 moving points [4] in order to evaluate the proposed algorithm. In each time interval, the points move at random speed and direction using a uniform distribution. The location of points in a square is considered to be 1000 x 1000 units, and τ_T is considered 20 generations for relocation of points. The moving speed of the points is from 1 to 10 m/s in the second period, 10 to 15 m/s in the third period, 15 to 25 m/s in the fourth period, and 25 to 35 m/s in the fifth period. The data generated is not a data stream, and has no limitations. Moreover, we will use the actual traffic control data presented in the papers [21, 22] as a moving dataset to evaluate the proposed algorithm. This dataset has a GPS route of 10,357 taxis from February 2 to February 8, 2008, in Beijing. In this experiment, on February 3, eight time intervals are considered. All implementations are carried out in the MATLAB software, and all the experiments are run on a computer with a 3.7 GHz Core i3 processor and 8 GB RAM.

4.1. DMOGA-DBSCAN evaluation

In this section, we compare the results obtained from the running of DMOGA-DBSCAN with the case where the values of the DBSCAN parameters are considered constant at different time intervals (MOGA-DBSCAN). Each algorithm is run 30 times on each dataset. In all experiments, the t-test is used as a statistical test in order to examine the comparison accuracy of the two evaluated algorithms. 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. A synthetic dataset is considered as the primary data to evaluate DMOGA-DBSCAN for the clustering of moving objects, and is generated by moving points in each time interval at random speed and direction using the uniform distribution. Additionally, the real data related to traffic control has been used as a moving dataset to evaluate the proposed algorithm. As the correct clustering result is unknown for moving data, the Outlier and Silhouette, internal cluster validity indices are used to assess clustering.

Table 1. t-test accepted hypothesis on synthetic dataset

Silhouette index						
		DMOGA-DBSCAN				
		T_1	T_2	T_3	T_4	T_5
MOGA-DBSCAN	T_1	H_0	-	-	-	-
	T_2	-	H_0	-	-	-
	T_3	-	-	H_1	-	-
	T_4	-	-	-	H_1	-
	T_5	-	-	-	-	H_1
Outlier index						
		DMOGA-DBSCAN				
MOGA-DBSCAN	T_1	H_0	-	-	-	-
	T_2	-	H_0	-	-	-
	T_3	-	-	H_1	-	-
	T_4	-	-	-	H_1	-
	T_5	-	-	-	-	H_1

In the case of moving data, it is supposed that after 20 generations, the environment changes for the first time in the second period (T2), then DMOGA-DBSCAN has 20 generations to generate the appropriate response set for the next period (T3), and this process continues. Using MOGA-DBSCAN for the initial dataset (without motion), the initial response set is generated, and the same initial response set is used for clustering at the next time intervals as the data moves.
According to Table 1 in the synthetic dataset, the t-test accepts the H1 hypothesis for the results of DMOGA-DBSCAN and MOGA-DBSCAN algorithms in the third to fifth time intervals.

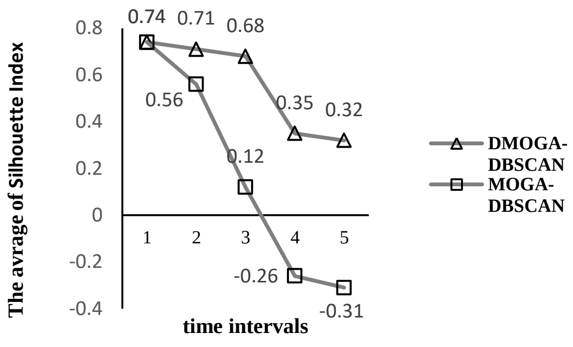


Figure 4. Average of Silhouette index in 30 runs on synthetic dataset.

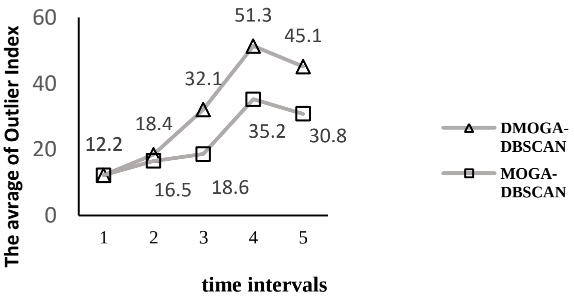


Figure 5. Average of Outlier index in 30 runs on synthetic dataset.

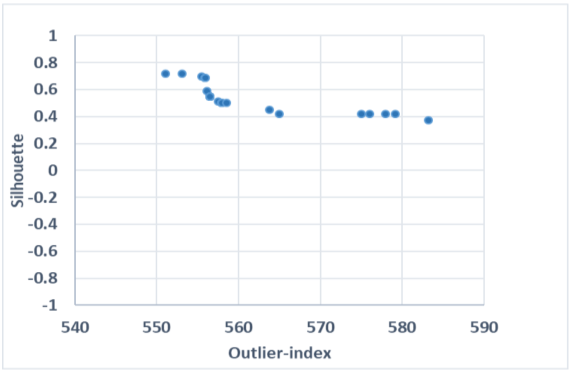
As Figure 4 indicates, in the third period, when the maximum speed is 1.5 times higher than the second period, based on the silhouette index, the performance of the DMOGA-DBSCAN algorithm decreases by 4% compared to the second period. Nonetheless, it performs 82% better than MOGA-DBSCAN. In the fourth interval, the maximum speed compared to the second time interval is 2.5 times, and based on the silhouette index, the performance of the DMOGA-DBSCAN algorithm is reduced by 50% compared to the second time interval. However, it has a 174% better performance than MOGA-DBSCAN. In the fifth interval, the maximum speed compared to the second time interval is 3.5 times, and based on the silhouette index, the performance of the DMOGA-DBSCAN algorithm is reduced by 55% compared to the second time interval. However, it has 197% better performance than MOGA-DBSCAN.

Based on Figure 5, the DMOGA-DBSCAN algorithm has 42%, 31%, and 32% better performance in detecting outliers than MOGA-DBSCAN in the third to fifth periods, respectively, in terms of the outlier index. Based on Figures 4 and 5 in the synthetic dataset, the results of the DMOGA-DBSCAN algorithm are 119% and 29% better than MOGA-DBSCAN in terms of the silhouette and outlier indices in different periods, respectively. As the results of the experiments show, after the second period, when the intensity of environmental changes increases, the performance of DMOGA-DBSCAN decreases slightly. However, it performs better than MOGA-DBSCAN.

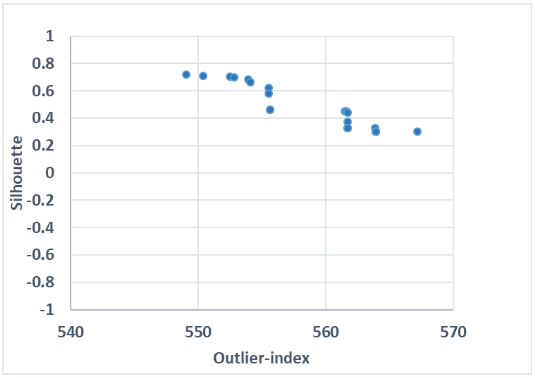
Based on Table 2 in the traffic dataset, the t-test for the results of DMOGA-DBSCAN and MOGA-DBSCAN algorithms accepts the null hypothesis at 10 and 18 o'clock, and the null hypothesis is rejected at the remaining hours. The Pareto front of the DMOGA-DBSCAN algorithm performed on the traffic dataset related to one of the best outputs of the DMOGA-DBSCAN shown in Figure 6. According to Figures 7 and 8 in the traffic dataset, the results of the DMOGA-DBSCAN algorithm are 24% and 8% better than MOGA-DBSCAN in terms of the silhouette and outlier indices in various periods, respectively. As the results of the experiments show, the displacement of the points changes the density of the clusters and changes the environment, and the MOGA-DBSCAN, which is performed with the initial parameters, does not have the necessary efficiency. In the third period and after that, when the environment changes more, DMOGA-DBSCAN performs better than MOGA-DBSCAN. Therefore, the experiments show that it is necessary to re-define the parameters in dynamic environments. It should be noted that the proposed algorithm is based on the DBSCAN algorithm, and its efficiency decreases in environments where there are multi-density clusters. Thus the proposed algorithm should be developed to determine appropriate values for the input parameters of the DBSCAN algorithm in multi-density environments.

Table 2. t-test accepted hypothesis on traffic dataset

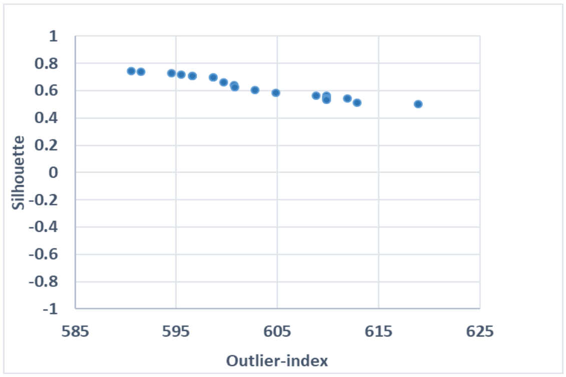
					Silhouette index				
					DMOGA-DBSCAN				
		8 o'clock (T_1)	**10 o'clock (T_2)**	**12 o'clock (T_3)**	**14 o'clock (T_4)**	**16 o'clock (T_5)**	**18 o'clock (T_6)**	**20 o'clock (T_7)**	**22 o'clock (T_8)**
MOGA-DBSCAN	**T_1**	H_0	-	-	-	-			
	T_2	-	H_0	-	-	-	-	-	-
	T_3	-	-	H_1	-	-	-	-	-
	T_4	-	-	-	H_1	-	-	-	-
	T_5	-	-	-	-	H_1	-	-	-
	T_6	-	-	-	-	-	H_0	-	-
	T_7	-	-	-	-	-	-	H_1	-
	T_8	-	-	-	-	-	-	-	H_1
					Outlier index				
					DMOGA-DBSCAN				
MOGA-DBSCAN	**T_1**	H_0	-	-	-	-			
	T_2	-	H_0	-	-	-	-	-	-
	T_3	-	-	H_1	-	-	-	-	-
	T_4	-	-	-	H_1	-	-	-	-
	T_5	-	-	-	-	H_1	-	-	-
	T_6	-	-	-	-	-	H_0	-	-
	T_7	-	-	-	-	-	-	H_1	-
	T_8	-	-	-	-	-	-	-	H_1



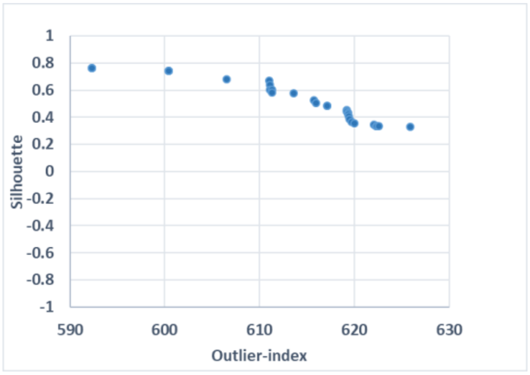
a. T_1



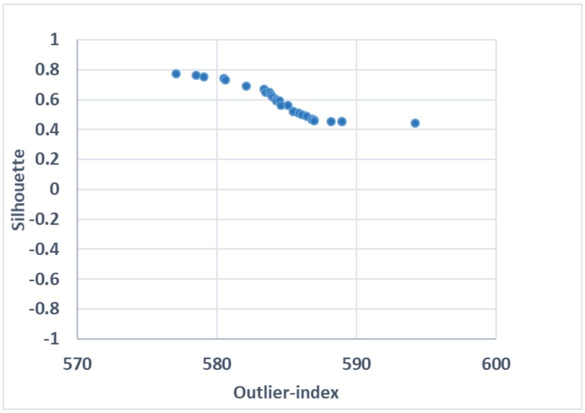
b. T_2



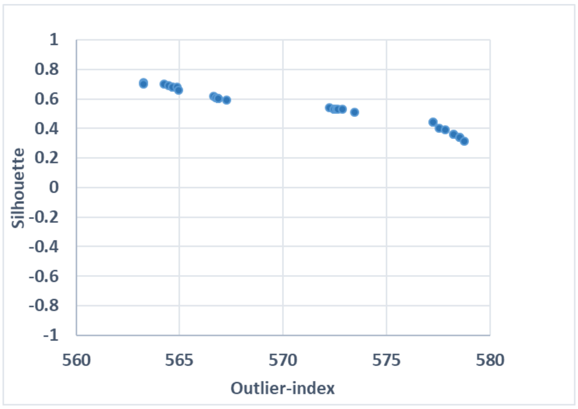
c. T_3



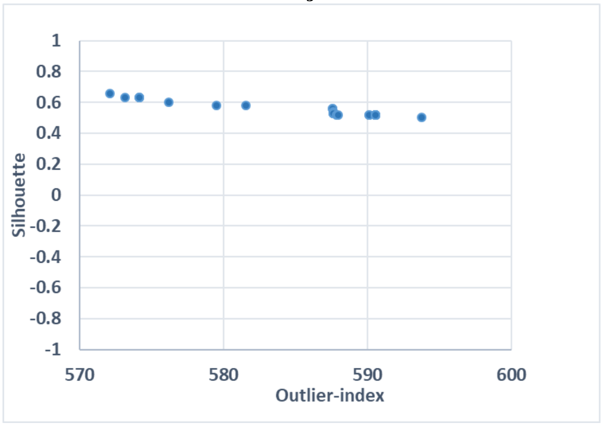
d. T_4



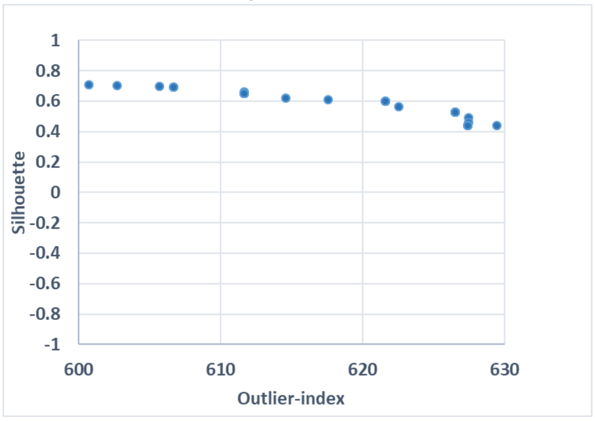
e. T_5



f. T_6



e. T_7



f. T_8

Figure 6. Pareto front of DMOGA-DBSCAN algorithm on traffic dataset.

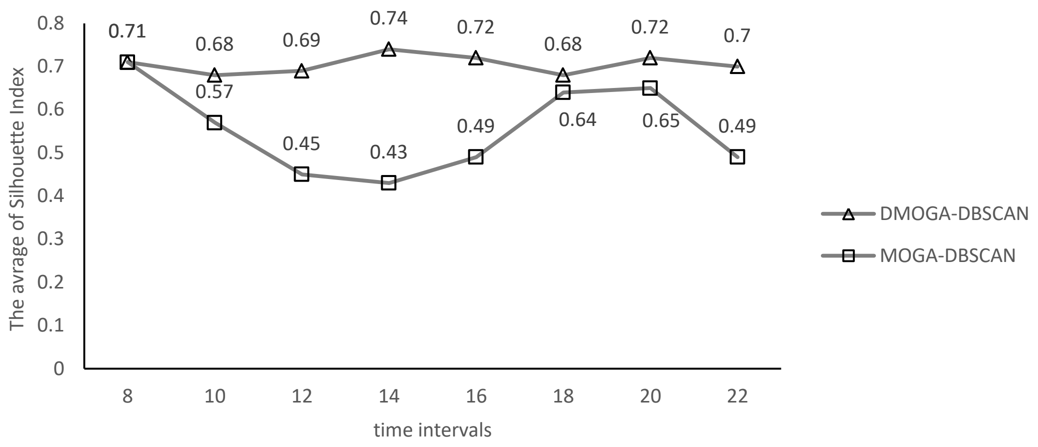


Figure 7. Average of Silhouettei in 30 runs on traffic datasets.

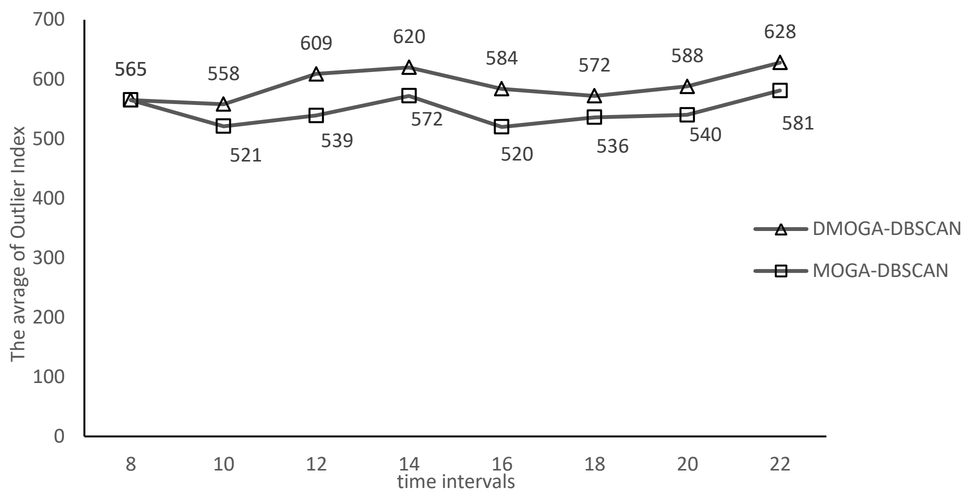


Figure 8. Average of Outlier index in 30 runs on traffic dataset.

5. Conclusion

In this paper, the DMOGA-DBSCAN algorithm has been introduced for clustering moving objects, which determines the parameters of the DBSCAN algorithm in accordance with the spatial variations of the points. The results obtained from running of DMOGA-DBSCAN on the synthetic datasets and traffic datasets were compared with the state where the values of the DBSCAN parameters at various intervals were considered constant (MOGA-DBSCAN). According to the results obtained in the synthetic dataset, the results of the DMOGA-DBSCAN algorithm are, respectively, 119% and 29% better than MOGA-DBSCAN in terms of the silhouette and the outlier indices at various intervals. Moreover, in the traffic dataset, the results of the DMOGA-DBSCAN algorithm are, respectively, 24% and 8% better than MOGA-DBSCAN in terms of the silhouette and the outlier indices at the various intervals. According to the results obtained, the DMOGA-DBSCAN algorithm outperforms MOGA-DBSCAN in terms of both the silhouette and the outlier indices.

For future studies, extending and developing the DMOGA-DBSCAN algorithm to dynamically determine the parameters of the DBSCAN algorithm in data clustering that has short response time and memory constraints is recommended.

References

[1] S. Kisilevich, F. Mansmann, M. Nanni, and S. Rinzivillo, "Spatio-temporal clustering," in *Data mining and knowledge discovery handbook*: Springer, 2009, pp. 855-874.

[2] Z. Falahiazar, A. BAGHERF, and M. Reshadi, "Determining the Parameters of DBSCAN Automatically Using the Multi-Objective Genetic Algorithm," *Journal of Information Science & Engineering,* vol. 37, no. 1, 2021.

[3] P. Kalnis, N. Mamoulis, and S. Bakiras, "On discovering moving clusters in spatio-temporal data," in *International Symposium on Spatial and Temporal Databases*, 2005, pp. 364-381: Springer.

[4] C. S. Jensen, D. Lin, and B. C. Ooi, "Continuous clustering of moving objects," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 9, 2007.

[5] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, and X. Xu, "Incremental clustering for mining in a data warehousing environment," in *VLDB*, 1998, vol. 98, pp. 323-333: Citeseer.

[6] N. Goyal, P. Goyal, K. Venkatramaiah, P. Deepak, and P. Sanoop, "An efficient density based incremental clustering algorithm in data warehousing environment," in *2009 International Conference on Computer Engineering and Applications, IPCSIT*, 2011, Vol. 2, pp. 482-486.

[7] A. M. Bakr, N. M. Ghanem, and M. A. Ismail, "Efficient incremental density-based algorithm for clustering large datasets," *Alexandria engineering journal*, Vol. 54, No. 4, pp. 1147-1154, 2015.

[8] P. Yadav and P. Sharma, "An Efficient Incremental Density based Clustering Algorithm Fused with Noise Removal and Outlier Labelling Technique," *Indian Journal of Science and Technology*, Vol. 9, No. 48, 2016.

[9] L. Pradeep and A.M. Sowjanya, "Multi-Density based Incremental Clustering" *International Journal of Computer Applications*, Vol. 116, No. 17, pp. 0975–8887, 2015.

[10] Y. Gong, R. O. Sinnott, and P. Rimba, "RT-DBSCAN: Real-Time Parallel Clustering of Spatio-Temporal Data Using Spark-Streaming," in *International Conference on Computational Science*, 2018, pp. 524-539: Springer.

[11] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Kdd*, 1996, Vol. 96, No. 34, pp. 226-231.

[12] L. Falahiazar, V. Seydi, and M. Mirzarezaee, "Sequential Multi-objective Genetic Algorithm," *Journal of AI and Data Mining*, vol. 9, no. 3, pp. 369-381, 2021.

[13] U. Maulik, S. Bandyopadhyay, and A. Mukhopadhyay, *Multiobjective Genetic Algorithms for Clustering: Applications in Data Mining and Bioinformatics*. Springer Science & Business Media, 2011.

[14] C. A. C. Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary algorithms for solving multi-objective problems*. Springer, 2007.

[15] 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, no. 2, pp. 182-197, 2002.

[16] B. Delaunay, "Sur la sphère vide," *Izvestia Akademii Nauk SSSR, Otdelenie Matematicheskikh i Estestvennykh Nauk*, pp. 793–800, 1934.

[17] P. Roy and J. Mandal, "A novel spatial fuzzy clustering using delaunay triangulation for large scale gis data (nsfcdt)," *Procedia Technology*, Vol. 6, pp. 452-459, 2012.

[18] K. M. Ramachandran and C. P. Tsokos, *Mathematical statistics with applications in R*. Elsevier, 2014.

[19] R. L. Ott and M. T. Longnecker, *An introduction to statistical methods and data analysis*. Nelson Education, 2015.

[20] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53-65, 1987.

[21] J. Yuan *et al.*, "T-drive: driving directions based on taxi trajectories," in *Proceedings of the 18th SIGSPATIAL International conference on advances in geographic information systems*, 2010, pp. 99-108: ACM.

[22] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 316-324: ACM.

تعیین پارامترهای الگوریتم DBSCAN در محیط‌های پویا به طور خودکار با بکارگیری الگوریتم بهینه سازی ژنتیک چند هدفه پویا

زینب فلاحی آذر[۱]، علیرضا باقری[۲]* و میدیا رشادی[۱]

[۱] گروه مهندسی برق و کامپیوتر، واحد علوم و تحقیقات دانشگاه آزاد اسلامی، تهران، ایران.

[۲] گروه مهندسی کامپیوتر ، دانشگاه صنعتی امیرکبیر، تهران، ایران.

ارسال ۲۰۲۱/۱۲/۲۰؛ بازنگری ۲۰۲۲/۰۴/۲۶؛ پذیرش ۲۰۲۲/۰۵/۳۱

چکیده:

خوشه بندی مکانی-زمانی حوزه فرعی نسبتاً جدیدی در داده‌کاوی است که محبوبیت زیادی به ویژه در دانش اطلاعات جغرافیایی به دست آورده است. یک نوع از داده مکانی-زمانی، اشیاء متحرک است که اطلاعات موجود درباره این اشیاء شامل آخرین موقعیت آنها می‌شود و دنباله موقعیت‌های قبلی نگهداری نمی‌شود. برای خوشه‌بندی اشیاء متحرک از استراتژی اجرای عمل خوشه‌بندی بر روی تمام دنباله‌های زمانی استفاده می‌شود. مشکل خوشه‌بندی مبتنی بر چگالی که از این استراتژی استفاده می‌کند این است که در هر بازه زمانی با توجه به تغییر مکان نقاط، چگالی خوشه‌ها نیز ممکن است تغییر نماید. بنابراین پارامترهای ورودی الگوریتمی مانند DBSCAN که برای خوشه بندی اشیاء متحرک بکار می‌رود نیز تغییر خواهند کرد و مجددا نیاز به تعیین آنها می‌باشد. روش‌هایی که تاکنون ارائه شده‌اند و از الگوریتم DBSCAN برای خوشه‌بندی اشیاء متحرک استفاده می‌کنند، پارامترهای ورودی DBSCAN را در طول زمان ثابت در نظر گرفته‌اند و راهکاری برای تعیین خودکار آنها ارائه نداده‌اند. برای رفع این مشکل در این مقاله از الگوریتم ژنتیک دو هدفه پویا جهت تعیین پارامترهای الگوریتم DBSCAN به صورت پویا و خودکار استفاده می‌گردد. از داده‌های واقعی مربوط به کنترل ترافیک شهر پکن به عنوان مجموعه داده متحرک جهت ارزیابی الگوریتم پیشنهادی استفاده شده است. نتایج آزمایشات نشان می دهد که بکارگیری الگوریتم پیشنهادی جهت تعیین پویای پارامترهای ورودی DBSCAN از نظر دو معیار Silhouette و Outlier عملکرد بهتری نسبت به اجرای DBSCAN با پارامترهای ورودی ثابت در طول زمان ایجاد می‌کند.

کلمات کلیدی: خوشه‌بندی مبتنی بر چگالی، الگوریتم DBSCAN، بهینه‌سازی پویا چند هدفه، خوشه‌بندی اشیاء متحرک، شاخص اعتبارسنجی خوشه‌ای.