# Density-based spatial clustering in the presence of obstacles

Amani A. Saad, Mohamed S. Abougabal and Amal Moustafa
*Computer Science and Automatic Control Dept., Faculty of Eng., Alexandria University, Alexandria, Egypt*

This paper proposes a density based clustering algorithm which takes obstacles into consideration. The proposed Clustering with Obstacle Entities algorithm COE-DBSCAN is based on the DBSCAN algorithm, which performs effective clustering for spatial data based on the notion of density. The paper presents a survey of existing spatial clustering algorithms and identifies the problems of related algorithms that deal with obstacles. The complexity of the proposed algorithm COE-DBSCAN is analyzed and some experimental work is presented. It is shown that COE-DBSCAN overcomes the problems of existing spatial clustering algorithms considering obstacles. The paper is concluded with some suggestions for further research issues.

مع وجود كم كبير من البيانات المخزنة في قواعد البيانات التى تحتوى على بيانات حيزية يفضل البعض إيجاد المجموعات من البيانات التي تشترك في خصائص متماثلة. ولذا يعد التصنيف الحيزي من أحد أهم عمليات التنقيب الحيزي للبيانات ولهذه الأهمية فقد أصبح هذا المجال موضوع بحث رائج. و بالرغم من استخدا م التصنيف الحيزي في العديد من التطبيقات الجغرافية والطبية وعلم الفلك وغيرها فأنه في أغلب خوارزميات التصنيف التي تم إعدادها حتى وقت اجراء هذا البحث فقد تم تجاهل وجود العوائق الطبيعية الموجودة في العالم الحقيقي، مثل الأنهار والبحيرات و الطرق السريعة مع أن وجودها يمكن أن يؤثر في نتائج التصنيف بصورة جوهرية. و لذا يتناول هذا البحث مشكلة التصنيف مع وجود عوائق و يتم فيه إقتراح خوارزيم قائم على الكثافة COE-DBSCAN وهو مبنى على الخوارزيم المعروف DBSCAN و يقوم بالتصنيف بكفاءة و فعالية مع أخذ عوامل الإعاقة في الإعتبار. وقد تم تحليل مدى تعقيد الخوارزيم المقترح كما تم عمل مقارنة للنتائج بالأعمال السابقة في هذا المجال. كما يقدم فى هذا البحث بعض التجارب العملية التى تمت لدراسة كفاءة الخوارزيم الخاص بالتصنيف الحيزى فى وجود عوائق ومدي تحقيقه للهدف المرجو منه وتمت مناقشة نتائج تلك التجارب. كما يضم ملخص للبحث وما تم أنجازه فيه وبعض الأقتراحات لإضافات مستقبلية.

**Keywords:** Spatial, Spatial data mining, Spatial clustering, Density based clustering, Clustering in the presence of obstacles

## 1. Introduction

Spatial data mining is a demanding field since huge amounts of spatial data has been collected in various applications, ranging from remote sensing to Geographical Information Systems (GIS), computer cartography, environmental assessment and planning. The collected data far exceeds people's ability to analyze it. Thus, new and efficient methods are needed to discover knowledge from large spatial databases. Different techniques such as generalization-based, clustering, spatial association rules, approximation and aggregation have been defined for mining of spatial data [1].

Spatial clustering, which groups similar spatial objects into classes,is an important component of spatial data mining. Due to its immense applications in various areas, spatial clustering has been a highly active topic in data mining research. Its application has been utilized in many fields [2].

Spatial databases contain spatial related information. Such databases include geographic (map) databases, VLSI chip design databases, and medical and satellite image databases. Spatial data mining is the discovery of interesting characteristics and patterns that may exist in large spatial databases. Clustering, in spatial data mining, is a useful technique for grouping a set of objects into classes or clusters such that objects within a cluster have high similarity among each other, but are dissimilar to objects in other clusters.

The spatial clustering methods are classified into five categories [3]:
- partitioning methods;
- hierarchical methods;
- density-based methods;
- grid-based methods; and
- constraint-based methods.

For the same data sets, the clusters discovered could be different according to the clustering methods used. For certain data set structures in certain applications like minefield detection (grouping mines in minefield), seismology (grouping earthquakes clustered along seismic faults) and astronomy (grouping stars in galaxies) there is a need for the density-based notion of a cluster, which can discover natural clusters with arbitrary shapes and give more accurate results than other methods.

The main reason that the clusters are recognized using the density based notion, is that within each cluster there is a typical density of points, which is considerably higher than outside of the cluster. Furthermore, the density within the areas of noise is lower than the density in any of the clusters. Thus, it can be concluded that density based algorithms could discover clusters with arbitrary shapes and are sensitive to noises and outliers.

On the other hand, it is found that in the real world, many physical obstacles exist such as rivers, lakes, mountains that their presence may affect the result of spatial clustering. Thus, it is useful to study the problem of spatial Clustering with Obstacle Entities (COE).

Although, the problem of COE in partitioning based methods [4] and in grid based methods has been studied [5], a density based algorithm, that considers the real life physical obstacles has not been proposed yet. Thus, for the importance of density based methods mentioned above, the problem of COE in density based algorithms needs to be investigated.

Indeed, there is a need for a novel density based algorithm that takes the obstacles into consideration and tries to overcome the drawbacks found in existing spatial clustering algorithms. Hence, the density based algorithm DBSCAN is selected as the framework, and the proposed algorithm is named COE-DBSCAN.

The rest of the paper is organized as follows. The related work is discussed in sections 2 and 3. The details of the proposed algorithm COE-DBSCAN is presented in section 4, the analysis of the complexity of the proposed algorithm is presented in section 5 . Some experimental results are presented in section 6. Moreover, section 7 presents a comparison of the algorithm results with other constraint-based methods. Finally, section 8 presents a summary of the paper, along with recommendations for future extensions.

## 2. Related work

Many studies have been conducted in cluster analysis. Table 1 summarizes Spatial Clustering methods in the literature. Constraint based methods in particular are discussed in section 3.

## 3. Constraint-based methods

The development of spatial data mining on large databases has provided many useful tools for the analysis of geographical data. However, most of these algorithms provide very few avenues for users to specify real life constraints which must be satisfied with the clustering. In many applications, physical obstacles like mountains and rivers could affect the result of clustering algorithms. So , a novel problem has been defined COE-Problem (clustering with obstacle entity). Two algorithms addressed this problem: COD-CLARANS [4], and SCPO [5]. These are briefly described below and summarized in table 2.

### *3.1. COD-CLARANS*

COD-CLARANS (clustering with obstructed distance based on CLARANS)[4] was the first clustering algorithm that takes into consideration the presence of obstacle entities. COD-CLARANS consists of two phases. The first phase breaks the database into several databases and summarizes them individually by grouping the objects in each sub-database in a micro-cluster. A micro-cluster is a group of points, which are so close together that they are likely to belong to the same cluster. The second phase is the clustering stage. The

Table 1
Spatial clustering methods

| Clustering methods | Examples | Descriptions | Remarks |
|---|---|---|---|
| Partitioning methods | K-means[6]<br>EM [7]<br>K-medoid[8] | - Cluster based on the notion of distance<br>- Use iterative reallocation techniques. | - K (Number of clusters) will be specified as input parameter.<br>- Cannot find arbitrarily shaped clusters. |
| Hierarchical methods | Agnes& Diana[8]<br>BIRCH [9]<br>CURE [10]<br>Chameleon [11] | - Create a hierarchical decomposition of the given set of data objects forming a tree, which splits the database recursively into smaller subsets. | - The Cluster merging and splitting operations are irreversible.<br>- Cannot swap objects between clusters. |
| Density-based methods | DBSCAN [12]<br>OPTICS [13]<br>DENCLUE [14] | - Cluster based on the notion of density | - Can find clusters with arbitrary shapes and sizes<br>- Sensitive to outliers |
| Grid-based methods | STING [15]<br>Wave-cluster [16]<br>CLIQUE [17] | - Use a grid data structure to decompose the area then label each cell as dense or non dense | - Fast processing time which is independent of the number of data objects<br>- Facilitate parallel processing |
| Constraint-based methods | COD-CLARANS [4]<br>SCPO [5] | - Extend any of the above techniques and<br>- Take the obstacles into consideration | - still in its infancy (only two algorithms in the literature) |

algorithm randomly selects k points as the centers for the required clusters and then tries to find better solutions. Since COD-CLARANS is an extension of the CLARANS algorithm, it suffers from similar drawbacks as CLARANS.

### *3.2. SCPO*

SCPO (Spatial Clustering in the Presence of Obstacles) [5] is a grid-based algorithm, which divides the spatial area into m, rectangular cells of equal areas. Then the algorithm labels each cell as dense or non-dense according to the number of points in that cell and an input threshold. The algorithm also labels each cell as obstructed (i.e. intersects any obstacle) or non-obstructed. The algorithm finds maximal connected regions of dense, non-obstructed cells. Then the algorithm finds a center for each obtained region.

A close study of table 2 reveals that although there are trials for developing spatial clustering algorithms with obstacles, yet there is no density based algorithm, that takes the real life obstacles into consideration. It can be concluded that the problems that are found in existing spatial clustering algorithms with obstacles that arise from their nature are as follows:

Table 2
Constraint-based spatial clustering algorithms

| Algorithm name | Description | Remarks |
|---|---|---|
| COD-CLARANS | Extend CLARANS (Partitioning based) | - Needs two input parameters<br>- Can not handle outliers<br>- Can not find arbitrary shaped clusters |
| SCPO | (Grid based) | - Needs two input parameters |

1. do not detect noises and outliers;(Outliers refer to spatial objects, which are not ontained in any cluster and should be discarded during the mining process);
2. do not detect clusters with arbitrary shpes and sizes;
3. need to know the number of clusters in advance; and
4. use randomize search and micro clustering in the preprocessing stage.

Indeed, it can be seen from the previous discussion, that there is a need for a novel spatial clustering algorithm with obstacles that overcomes all the drawbacks mentioned above. Thus, a density based algorithm COE-DBSCAN which is based on the DBSCAN algorithm is proposed in the next section.

## 4. The proposed COE - DBSCAN algorithm

Most clustering algorithms assume direct Euclidean distance among the objects to be clustered without obstacles in the way. However, most applications do have obstacles in presence, and the omission of such obstacles may lead to distorted and often useless clustering results. Hence, the term *obstructed distance* is used to represent the distance between two points in the presence of obstacles.

During the clustering phase the COE-DBSCAN algorithm needs to select the Eps-neighborhood; using the obstructed distance and not the Euclidean distance of each point; where Eps is a threshold taken as an input parameter.

The proposed COE-DBSCAN algorithm consists of two phases; a *pre-processing phase* that materializes information which facilitates the obstructed distance computation, and the *clustering phase* which performs the actual clustering. These two phases are described below:

- Phase I is the Pre-processing Stage. In this phase a *Binary Space Partitioning tree (BSP)* [18] is constructed as well as a complete *Visibility Graph* for later use in computing the obstructed distance function.
- Phase II is the Clustering stage. In this phase the DBSCAN algorithm is applied, but instead of selecting the Eps-neighborhood for each data object based on the Euclidean distance, the selection is based on the obstructed distance computed in phase I.

The following definitions are needed before the algorithm is presented.

*Definition 1*: (*Eps*- neighborhood of a point) The *Eps*- neighborhood of a point *p*, denoted by: $N_{Eps}(p)$, is defined by $N_{Eps}(p) = \{q \in D \mid \text{dist}(p,q) \leq Eps\}$ [12].

*Definition 2*: (Directly density reachable) A point *p* is directly density reachable from point *q*, with respect to, *Eps* and MinPts if $p \in N_{Eps}(q)$ and $| N_{Eps}(q) | \geq$ MinPts (core point condition) where, *MinPts* is the minimum number of points in an Eps-neighborhood of a point in order to be considered to be in a cluster and not an outlier.

*Definition 3*: (Density reachable) A point *p* is density reachable from point *q*, with respect to, *Eps* and MinPts if there is a chain of points $p_1,\ldots p_n$, $p_1 = p, \ldots p_n = q$, such that $p_{i+1}$ is directly density reachable from $p_i$.

### 4.1. Phase I: The preprocessing stage

The aim of this phase is to complete as much pre-calculations as possible so that the future procedures do not have to repeat the same operations. All information from this stage is stored in memory. During this phase, the BSP tree (Binary Space Partitioning Tree) for the obstacles defined in a region R, and the visibility graph are built, in order to be used in calculating the obstructed distance later on.

#### 4.1.1. The BSP tree

The Binary Space Partitioning (BSP) Tree [18] is a data structure which can efficiently determine whether two points *p* and *q* are visible to each other within the region R. The point *p* is defined to be visible from *q* in the region *R* if the straight line joining p and q does not intersect any obstacles.

In the algorithm the BSP tree is used to determine the set of all visible obstacle vertices from a point p. The notation *vis(p)* is used to denote such a set of vertices.

The Binary-Space-Partitioning (BSP) tree has been widely-used and studied in the graphics and computer games industry. The advantage of this data structure is that it can be used to efficiently determine a data object location in relation to the polygonal obstacles within the space. With this information, the data space can be further processed and the visibility from that data object can be discovered.

#### 4.1.2. Visibility graph

A visibility graph is a graph whose set of nodes *V* correspond to vertices of the polygonal obstacles, and whose set of edges *E* correspond to the pairs of vertices that are mutually visible to each other. In an obstacles planar space, there is an infinite number of paths between any two points *p* and *s*. The shortest path is composed of segments whose endpoints are either *p, s*, or the vertices of the polygons in the planar space [19].

Given a set of $m$ polygonal obstacles, $O=\{O_1,..., O_m\}$, the visibility graph is a graph $VG=(V,E)$ such that each vertex of the obstacles has a corresponding node in $V$, and two nodes $v_1$ and $v_2$ in $V$ are joined by an edge in E if and only if the corresponding vertices they represent are visible to each other.

To generate VG, the BSP-tree is used, and a search for all visible vertices from each vertex of the obstacles is conducted. The visibility graph is pre-computed because it is used for finding the obstructed distance between any two points in the region.

Finding all polygon vertices that are visible to a point is achieved in two main steps as follows; obtain a priority list from the BSP-tree with respect to the point, and then incrementally compute the point's visibility.

*4.1.3. Computing the obstructed distance*

In order to compute the obstructed distance, the shortest path between all obstacles' vertices must be materialized. Hence, the Dijkstra algorithm is applied for this purpose. Dijkstra algorithm is also called the single-source shortest path algorithm [20]. The algorithm discovers the shortest paths between a given single source to all vertices in a weighted graph. The shortest path from s to $p$ is defined as a path from s to p such that the sum of the weights of the arcs on the path is minimized. Thus the algorithm is applied on the visibility graph.

*4.2. Phase II: clustering stage*

In this section, the proposed COE-DBSCAN algorithm is discussed and its pseudo code is presented. It is based on the DBSCAN clustering algorithm but takes the obstacles into consideration. In addition, the obstructed distance computation between data objects is presented.

*4.2.1. COE-DBSCAN algorithm*

To find clusters, COE-DBSCAN starts with an arbitrary point $p$ and retrieves all points *density reachable* from $p$ with respect to *Eps*, the *Eps* neighborhood data objects to point $p$.

If $p$ is a *core point*, the number of *Eps* neighbors data objects exceeds the minimum number of points MinPts accepted as an input parameter, so, this procedure yields a cluster and only the visible neighbors to the point p or the objects with obstructed distance less than *Eps* are added to this cluster. If $p$ is a *border point*, no points are density reachable from $p$, so COE-DBSCAN visits the next point in the database.

*Algorithm: COE-DBSCAN*
*Input:*
1. A set of n objects $\{P_1,P_2,.....P_n\}$ and a set of polygon obstacles$\{O_1,O_2,.....O_m\}$ in a spatial area S.
*2. Eps*, MinPts are the global density parameters, which represent the minimum number of points in the Eps neighborhood.
*Output:* Clusters in S along with their labels
*Method:*
Function COE-DBSCAN
Precondition: All objects in S are unclassified
ClusterId = 1 (construct a new cluster);
For ($i=1$; $i<n$ ; $i$ ++)
{
Determine the $i^{th}$ data object $P_i$ from the $N$ objects;
If the cluster of $P_i$ is unclassified Then
//call function Expand Cluster to construct a cluster containing $P_i$
Expand Cluster returns True if it detects all data objects that belong to cluster Clustid (reachable from $P_i$). It returns False if $P_i$ is still unclassified //
If ExpandCluster ($N$ objects, $P_i$, ClusterId, *Eps*, MinPts) Then
{
ClusterId = nextId (construct a new cluster);
}
}
Output the clusters with their label;

The following pseudo code illustrates the ExpandCluster function to find all reachable data objects from a certain one.

*Function: ExpandCluster*
*Input:*

1. A set of $N$ objects $\{P_1,P_2,...P_n\}$ and a set of polygon obstacles $\{O_1,O_2,...O_m\}$ in a spatial area S.
2. The $i^{th}$ data object $P_i$ and the cluster ClustId.
*3. Eps*, MinPts are the global density parameters, which represent the minimum number of points in the *Eps* neighborhood.
*Method:*

```
Retrieve the Eps neighborhood N Eps (Pi) of the object;
If  N Eps(Pi).size < MinPts
Then   // Pi is not a core point//
  {
    Mark  Pi as  noise
    Return False;
  }
Else          // Pi  is a core point//
  {
Mark Pi with ClustId
Push all objects From N Eps( Pi) in a list,Q;
//The temporary list Q contains all objects that are candidates to be in the same cluster as Pi //
While Q is not empty Do
    {
current := Q.first();
// Take the first element in the list Q and detect if this current element is visible to Pi, i.e., the line between the current element and Pi does not intersect with any obstacle. If not compute the obstructed distance between Pi and current d' (current, Pi )//
IF  (current object is visible to Pi)  OR (d'(current, Pi) < =Eps)
Then
 {Mark current with ClustId;
 Retrieve all objects in N Eps(current) ;
  //the objects not yet classified or are marked as noise//
IF  N Eps(current).size >= MinPts
Then
    {
    For (j=1 ;j<N Eps(current).size ; j++)
                    {
Push the unclassified objects that are visible to the current element OR the obstructed distance between the current element and these objects is less than Eps in the list Q
Mark these objects with the current cluster ClustId;
            }
 }End if;
}
Else
{
  Mark current as unclassified
} End if;
 Q.delete;
 }End While     // Q list is not empty//
Return True;
} End if;
```

It must be noted that the ClusterId of points which have been marked to be NOISE may be changed later, if they are density -reachable from some other point in the database. This happens for border points of a cluster. Those points are not added to the list because it is already known that a point with a ClusterId of NOISE is not a core point. Adding those points to the list would only result in additional region queries which would yield no new answers. If two clusters *C1* and *C2* are very close to each other, it might happen that some point *p* belongs to both, *C1* and *C2*. Then *p* must be a border point in both clusters because otherwise *C1* would be equal to *C2* since global parameters are used. In this case, point p will be assigned to the cluster discovered first. Except from these rare situations, the result of COE-DBSCAN is independent of the order in which the points in the database are visited.

COE-DBSCAN needs two parameters, *Eps* and MinPts. However, the experiments indicate that the k-dist graphs for $k > 4$ do not significantly differ from the 4-dist graph and, furthermore, they need considerably more computations. Therefore, parameter MinPts will be eliminated by setting it to 4 for all databases (for 2-dimensional data).

To compute the obstructed distance in the Expand Cluster algorithm, it takes two phases as follows:
*Phase I*: compute an index *VV* entry for any pair of obstacle vertices.

The materialization of this index is equivalent to finding the all pairs shortest path in the visibility graph *VG*. It can be shown that the computation of the shortest path between any two invisible points in R will often require the calculation of the obstructed distance between vertices. As such materializing the *VV* index will avoid the redundant computation of

these distances. And this will be done by applying the Dijkstra algorithm [21].

*Phase II*: for any two points *p* and *q*, for which the obstructed distance needs to be computed, first we have to determine the set of all visible obstacle vertices from p and q using the BSP tree. These sets will be denoted as *vis*(*p*) and *vis*(*q*). Then $v_i$ is selected from *vis*(*p*) and $v_j$ is selected from *vis*(*q*) such that the distances $d(vi,v_j)$ is minimum. Thus, the obstructed distance between the two points *p, q* denoted *d'*(*p,q*) will be computed as follows:

Obstructed distance $d'(p,q) = d(p,v_i) + d(v_i,v_j) + d(v_j,q)$

Once phase I is completed, the execution of phase II is trivial except for the formation of *vis*(*p*), and *vis*(*q*) with respect to point *p* and *q*. The BSP tree will be used for this purpose.

## 5. Complexity analysis of the proposed algorithm

The complexity of COE-DBSCAN will be analyzed in this section. The following notations are defined for this discussion.

*N* is the number of data objects in database *D*,
*G* is the visibility graph with a set of vertices,
*V* is the and a set of edges *E* that connect pairs of vertices that are mutually visible,
$|V|$ is the number of vertices in *G*, and
$|E|$ is the number of edges in *G*.

The discussion is separated into two parts: Phase I and Phase II of COE-DBSCAN as described in the previous section.

### *5.1. Complexity analysis of phase I*

This phase complexity is divided into three major steps: the complexity of building the binary space partitioning (BSP) tree, the complexity of building the visibility graph, and the complexity of finding the shortest path between the obstacles' vertices using Dijkstra's algorithm.

#### *5.1.1. Complexity analysis of the BSP tree construction*

Both the BSP-tree construction and the visibility checking method depend on many variables. These include the number of obstacles, the number of edges in each obstacle, the distribution of the polygons, and also the location of the interested viewpoint. In order to analyze the complexity of the BSP-tree construction, we need to consider the worst-case scenario. Suppose there are a set of polygons with *V* edges in total (note that, the number of edges in any polygon is equal to the number of its vertices). One polygon edge is selected to construct the hyper-plane in every partition. In the worst case, each hyper-plane splits all edges in its subspace. Consequently the complexity is $O(|V|^2)$.

#### *5.1.2. Complexity analysis of the visibility graph construction*

To construct the visibility graph, as discussed in the previous section, each obstacle vertex V must be examined, and the BSP-tree is queried for all other vertices that are visible to *V*. The query takes $O(|V|)$ time, therefore the running time to construct G is $O(|V|^2)$.

#### *5.1.3. Complexity analysis of computing the obstructed distance*

The total expected running time for Dijkstra's algorithm [9] is $c_1 |V| \log |V| + c_2 |E| \log |V|$ which is nearly $O(|E| \log |V|)$.The performance of the Dijkstra's algorithm depends on the nature of the weighted graph. When *G* is a dense graph, the simple array implementation produces better results than the binary heap implementation. The running time for these three steps will be:
$O(|V|^2) + O(|V|^2) + O(|E| \log |V|)$ which is nearly $O(|V|^2 + |E| \log |V|)$.

### *5.2. Complexity analysis of phase II*

Phase II consists of applying the DBSCAN algorithm by using the obstructed distance neighborhood, computed in phase I, instead of the Euclidean distance. The complexity of phase II of COE-DBSCAN will refer to the same concept as the complexity of DBSCAN. In the worst-case the function ExpandCluster, which is applied to define all data objects for a certain cluster, will be implemented N time for each data point. So, the complexity of Expand Cluster will depend on the call of $Eps(P_i)$ which returns the *Eps* neighborhood, of the point

depending on obstructed distance, not the Euclidean distance.

Since, region queries can be supported efficiently by spatial access methods such as R* trees [22] which are assumed to be available in a spatial database system for efficient processing of several types of spatial queries. The height of an *R** tree is $O$ (log $N$) for a database of $N$ points in the worst case and a query with a small query region has to traverse only a limited number of paths in R*-tree. Since the *Eps*-Neighborhoods, are expected to be small compared to the size of the whole data space, the average run time complexity of a single region query is $O$ (log $N$). For each of the $N$ points of the databases, we have at most one region query. Thus, the average run time complexity of COE-DBSCAN is $O$ ($N$ log $N$ ).

*5.3. Total complexity analysis*

The total complexity of the COE-DBSCAN algorithm is the sum of expected running time of Phase I and Phase II:

$O(PHASEI) = \{ O(|V|^2 + |E| \log |V|) \}$.

$O(PHASEII) = \{O(N \log(N)) \}$.

Since $|V|$ the number of vertices, is assumed to be far less than the number of data objects $N$. Thus, the total complexity of COE-DBSCN is $O(N \log(N))$.

## 6. Experimental results

To assess the validity of the proposed algorithm, experiments were conducted using synthetic data. The screenshots of the clustering results generated by DBSCAN were compared to the clusters discovered by COE-DBSCAN. The data was generated by first placing simple polygons in the region R. Then, data points were randomly generated on the region R, and the data points that reside inside of the polygons, were eliminated. The polygons in the planer space represent the obstacles, and the points represent the data objects to be clustered.

In this section three different data sets containing points in two dimensions are selected in order to demonstrate the main features of the proposed algorithm. The complete set of the experiments that were conducted is presented in [23].

*6.1. Data set I*

First, a simple data set is selected in order to prove that the proposed algorithm takes the obstacles into consideration. It has only one cluster with three obstacles. The screenshots of the COE-DBSCAN steps will be shown in the following figures. In phase I, the original data set is shown in fig 1-a, the visibility graph of the obstacle vertices and all pairs shortest path in the visibility graph using Dijkstra's algorithm are shown in fig. 1-b. In phase II the clusters discovered by COE-DBSCAN are presented in fig. 1-c. Then, the DBSCAN clustering results are displayed in fig. 1-d.

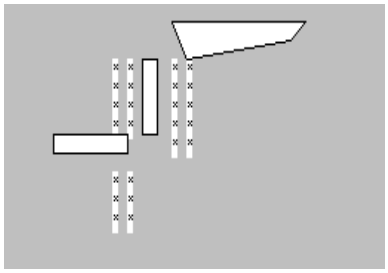
The COE-DBSCAN pre-processing steps can be summarized as follows:

- select one of the obstacle vertices;
- determine the vertices visible to the selected one;
- draw an edge in between;
- compute the distance of this edge to be used in Dijkstra's algorithm; then
- repeat the above process for all obstacle vertices.

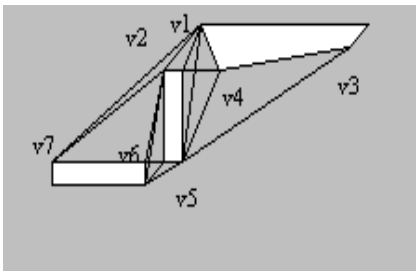
It is observed that the result of the DB-SCAN algorithm in fig. 1-c is one cluster which contains the obstacles between its objects. On the other hand, fig. 1.d shows that the proposed algorithm COE-DBSCAN produces three different clusters split by the obstacles. Thus COE-DBSCAN solves the problem that it was developed for. The next two experiments are presented in order to show other important features of COE-DBSCAN.

*6.2. Data set II*

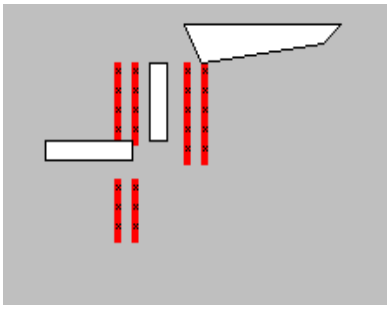
This data set is selected in order to express the main feature of COE-DBSCAN, which is the detection of noises and outliers. Thus, for different shapes of obstacles with noises and outliers data objects, fig. 2-a illustrates the original data set. Fig. 2-b represents the obstacle visibility graph.



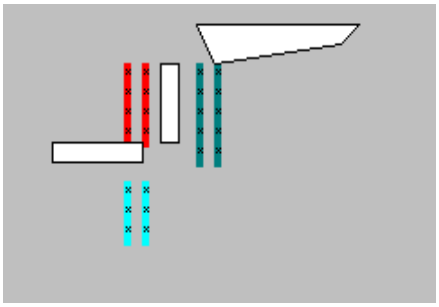
(a) Data Set I (Original Data)




(b) Data Set I (Visibility Graph)



(c) Data set I (DBSCAN Results)



(d) Data Set I (COE-DBSCAN

Fig.1. Data set I.

By observing fig. 2-c and fig. 2-d, it is observed that the outliers *n1*, *n2* and *n3* have been detected by both algorithms DBSCAN and COE-DBSCAN. This is due to the nature of both algorithms which are density-based, thus they have the ability to detect noise and outliers. However, the number of clusters found by DBSCAN for this data set is five clusters with a number of noise and outlier data objects, while COE-DBSCAN discovered ten clusters, and also had the ability to discover noise and outliers.

*6.3. Data set III*

This data set is selected in order to express an important feature of COE-DBSCAN, which is the detection of data clusters with different sizes and shapes.

By observing fig. 3-a, it can be seen that there are two rectangular clusters; *C1* is a small one while *C2* is a large one, and there are two randomly shaped clusters *C3* and *C4*.

Fig. 3-b displays the obstacle visibility graph, it is noticed that this visibility graph is the same as the visibility graph drawn for the previous data set as there is no change in the obstacle positions.

It is shown in fig. 3-c and fig. 3-d that because of their similar nature, both the DBSCAN algorithm and the COE-DBSCAN algorithms can detect these clusters with different shapes and sizes.

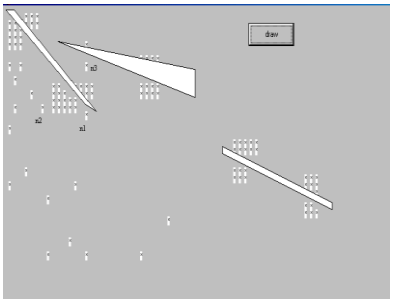
However, the number of clusters found by DBSCAN for this data set is five clusters as it did not sense the existence of obstacles, while, COE-DBSCAN discovered ten clusters.

Table 3 Comparison with other Constraint-based Algorithms.

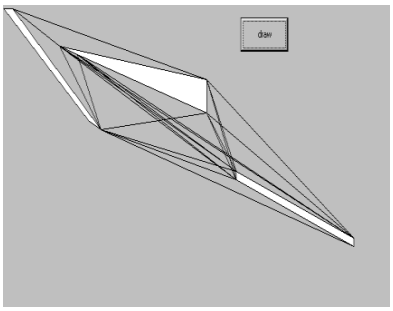
Thus, COE-DBSCAN represents a new efficient clustering algorithm which can handle outliers and noise. It is also capable of discovering clusters with arbitrary shapes and sizes and problem of input parameters is reduced which makes it less sensitive to the input from the user.

## 8. Conclusions and future work

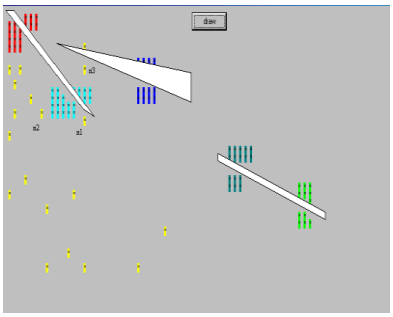
In this paper, an efficient spatial clustering algorithm COE-DBSCAN is proposed which takes the presence of obstacles into consideration. The proposed algorithm consists of two



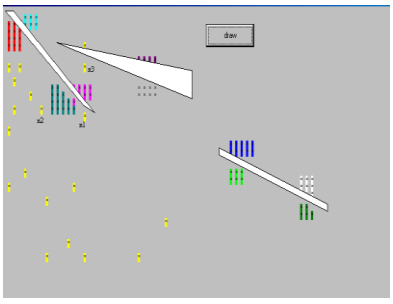
(a) Data Set II (Original Data)



(b) Data Set II (Visibility Graph)

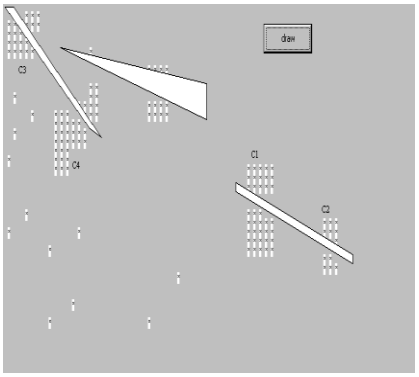


(c) Data Set II (DBSCAN Results)

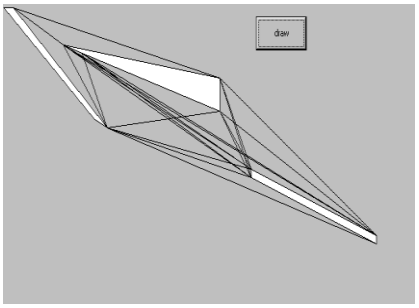


(d) Data Set IV (COE-DBSCAN Results)

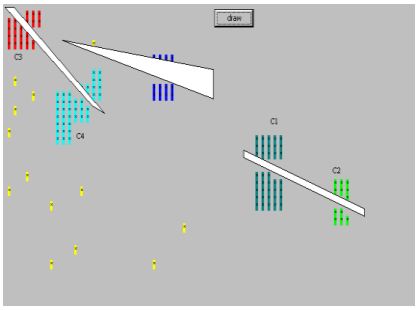
Fig. 2. Data set II.



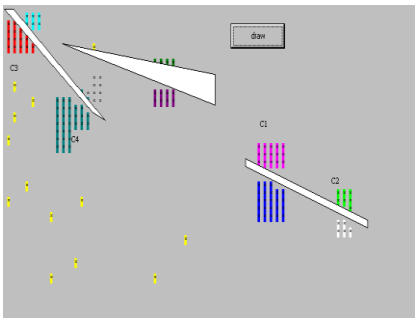
(a)Data Set III (Original Data)



(b) Data set II (original Data)



(c) Data set III (Visibility Graph)



(d) Data set III (COE-DBSCAN Results)

Fig. 3. Data Set III.

Table 3
Comparison with other constraint-based algorithms

| | COD-CLARANS | SCPO | COE-DBSCAN |
|---|---|---|---|
| Complexity | $O(m\|V\|) + O(m^2\|V\|)+ O(N\|V\|)$ (where m is the number of micro clusters) | $O(N) + O(k^2\|V\|) + O(k^3\|V\|)$. (where k is the number of space divisions) | O(Nlog(N)) |
| The number of input parameters | Two input Parameters | Two input parameters | Two input parameters (one of them is set to 4) |
| Handling outliers. | Can not handle outliers or noise | Handles outliers and noise | Handles outliers and noise |
| Clusters discovered | Discovers clusters with same shape and size | Discovers clusters with rectangular shape and different sizes | Discovers clusters with arbitrary shapes and sizes |
| Method used | Based on randomize search | Not based on randomize search | Not based on randomize search |
| Scale of data objects | Can not handle a large number of data objects. It needs a pre-clustering stage | Can handle a large number of data objects | Can handle a large number of data objects |

main stages, the preprocessing stage in which the visible neighborhood for each data object is determined using the BSP tree, and the clustering stage where the COE-DBSCAN algorithm is applied on the visible neighborhood only. As the proposed algorithm is an extension of a density-based algorithm DBSCAN, so it has the following advantages which have been verified by the experiments presented in this paper:
- it handles outliers and noise;
- it can discover clusters with different sizes and shapes;
Moreover,
- it finds the natural number of clusters in the spatial area without the need to know the number of clusters in advance; and it does not need a micro clustering stage during its preprocessing.

Therefore, the COE-DBSCAN algorithm eliminates the problems outlined in this paper for existing related algorithms COD-CLARANS and SCPO.

Indeed, Clustering in the presence of Obstacle Entities (COE) has presented a novel and interesting problem in the field of data mining. However, since the algorithm proposed COE-DBSCAN is an extention of the DBSCAN algorithm, so it will suffer from similar drawbacks as DBSCAN. Work can be done to improve the application of COE-DBSCAN.

Some of the future work is suggested as follows:

COE-DBSCAN requires two input parameters Eps (the neighborhood distance) and MinPts(the minimun no of points). The parameter MinPts has been eliminated by setting it to 4 in [12]. It has been shown in [14] that it is hard to determine a suitable value for Eps also some special cluster structures can not be revealed with the use of global density parameters. So, there is a need to investigate the implementation of the pre-processing stage proposed in this paper to the Optics: Ordering Points to Identify the Clustering Structure algorithm, which is an extention to DBSCAN. A similar study has been conducted in [24]. Furthermore, the preprocessing stage proposed can be implemented for other density-based algorithms like GDBSCAN [25].

There are two scalability issues raised by the proposed algorithm. The first issue is the number of vertices in the obstacles (V) compared with the number of data points to be clustered (N). The current COE-DBSCAN algorithm is based on the assumption that the number of vertices in the obstacles V is far less than the number of the data objects N. So, future research has to be done to investigate the cases where the number of obstacles is near the number of data objects.

The second issue is the data objects' type. As, the type used in the proposed algorithm is the point in 2-dimensions only. So, We believe that the research considering other spatial types like lines or polygons, and other

dimension, will be an interesting and practical research topic for future study.

While physical obstacles provide one type of constraints, operational requirements provide another. In the real world, the clustering task needs to comply with given rules and conditions. Indeed, besides constraints involved with obstacle entities, there are other interesting constraints that may require changing the nature and the emphasis of the clustering algorithms.

**Acknowledgements**

The authors are grateful to the valuable comments made by the reviewers, which added to the value of this paper.

**References**

[1] Krystof Koperski, "Spatial Data Mining: Progress and Challenges," Proc. SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, and Technical Report 96-08, University of British Columbia, Vancouver, Canada (1996).

[2] T. Raymond Ng, Jiawei Han, "Efficient and Effective Clustering Methods for Spatial Data Mining," (1994), Proc., 20th International Conference on Very Large Data Bases , Santiago, Chile, pp. 144-155 (1994).

[3] Jiawei Han, Micheline Kamber and K.H. Anthony Tung, "Spatial Clustering Methods in Data Mining: A Survey," 1st Conf on Geographic Data Mining and Knowledge Discovery, Taylor and Francis (2001).

[4] K.H. Tung, J. Hou, and J. Han, "Spatial Clustering in the Presence of Obstacles," Proc. 2001 Int. Conf. on Data Engineering (ICDE'01), Heidelberg, Germany (2001).

[5] E. El Mohamed -Sharkawi, Mohamed A. EL-Zawawy", Algorithm for Spatial Clustering with Obstacles". In Proceedings of first International Conference on Intelligent Computing and Information Systems (ICICIS 2002), Cairo, Egypt, pp. 249-254 (2002).

[6] J. Mac Queen. "Some Methods for Classification and Analysis of Multivariate Observations," Proc. 5th Berkeley Symp. Math., Statist., pp. 281-297 (1967).

[7] D.Yu., S. Chattenjou, C. Sheikhoeleslami, A.Zhang, "Efficiently Detecting Arbitrary Shaped Clusters in Very Large Data sets with High Dimensions," SUNY Buffalo, Computer Science Technical Report 98-08 (1998).

[8] U. Kaufman, and P.J. Rousseuw, Finding Groups in data: an Introduction to Cluster Analysis. John Wiley and sons publishers (1990).

[9] T. Zhang, R. Ramakrishnan, and M. Lvny. "Birch: an Efficient Data Clustering Method for Very Large Databases." Proc. 1996 ACM SIGMOD Conf, pp. 103-114 (1996).

[10] S. Guha, R. Rastagi, and K. Shim. "CURE: An Efficient Clustering Algorithm for Large Databases," Proc. 1998 ACM SIGMOD Conf., pp. 73-84 (1998).

[11] G. Karypis. E.H. Han, and V.Kumar. "Chameleon: A Hierarchical Clustering Algorithm Using Dynamic Modeling." Computer , Vol. 32, pp. 68-75 (1999).

[12] M. Ester, H.P. Kriegel, J. Sander and X. Xu. "A Density Based Algorithm for Discovering Clusters in Llarge Spatial Databases with Noise," In Proceedings of the 2nd International. Conference on Knowledge Discovery and Data Mining, 226-231, Portland, Oregon, USA (1996).

[13] M. Ankerst, M. Breunig, H.P. Kriegel, and J. Sander, "OPTICS: Ordering Points to Identify the Clustering Structure," In Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, 49-60, Philadephia Pennsylvania, USA (1999).

[14] A. Hinneburg, and D.A. Keim. "An Efficient Approach to Clustering in Large Multimedia Databases with Noise," Proc. Int. Conf. Knowledge Discovery and Data Mining (KDD'98), pp. 58-65 (1998).

[15] W. Wang, I. Yang, and R. Muntz. "STING: A Statistical Information Grid Approach to Spatial Data Mining," Proc. 1997 Int. Conf. Very Large Databases (VLDB'97), pp. 186- 195 (1997).

[16] G. Sheikhoeleslami, S. Chattenjee, A. Zhang. "Wave Cluster:A Multi Resolution Clustering Approach for Very Large

Spatial Databases", Proc 1998 Int. Conf. Very Large Databases (VLDB'98), pp. 428-439 (1998).

[17] R. Agrawal, and R. Srikant. "Fast Algorithms for Mining Associations Rules," Proc. 1994 Int. Conf. Very Large Databases (VLDB'94), pp. 487-499 (1994).

[18] J.D. Foley, A.V. Dam, S.K., and Feiner, J.F. Hughes. Computer Graphics: Principles and Practice .Second Edition. Addison-Wesley Publishing Company (1992).

[19] J.O. Rouke. Computational Geometry in C, Second Edition., Cambridge, Uk: Cambridge University Press (1994).

[20] M Aaron, Tenenbaum, and M.J. Augenstein, Data Structure Using Pascal. Second Edition. Prentice-Hall International, Editions (1986).

[21] T. Carmen, C. Leiserson, and R. divest. Introduction to Algorithms. The Mit Press, Cambridge (1990).

[22] N. Beckmann, H.P. Kriegel, Schneider R, and Seeger B. 1990. "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles", Proc. ACM SIGMOD Int. Conf. on Management of Data, Atlantic City, NJ, pp. 322-331 (1990).

[23] Amal Mostafa, Density Based Spatial Clustering in the presence of Obstacles, Master of Science Thesis, Computer Science and Automatic Control Department, Faculty of Engineering, Alexandria University, (2003).

[24] A. Salem, T. EL-Areef, A. Hassanien and M.F. Khater. "Towards An Efficient Density-Based Clustering Of Spatial Data," In Proceedings of first International Conference on Intelligent Computing and Information Systems (ICICIS 2002),Cairo, Egypt, pp. 269-274 (2002).

[25] J. Sander, M. Ester, H.P. and X. XU Kriegel, "Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and its Applications," in: Data Mining and Knowledge Discovery, Kluwer Academic Publishers, 2, pp. 169-194 (1998).

Received October 10, 2004
Accepted March 7, 2005