NPClu: A Methodology for Clustering Non-point Objects

Maria Halkidi **Michalis Vazirgiannis** **Yannis Theodoridis**
Dept of Informatics
Athens University of Economics & Business
Patision 76, 10434, Athens, Greece (Hellas)
{mhalk, mvazirg}@aueb.gr

Data & Knowledge Engineering Group
Computer Technology Institute
Patras, Greece (Hellas)
ytheod@cti.gr

Abstract

Clustering is an important task in managing voluminous data so as to identify significant groups in an underlying data set and extract "interesting" knowledge from it. Since it is widely recognized as a major tool in a number of applications in many fields (business and science), a number of clustering techniques and algorithms have been proposed and are available in literature. The vast majority of algorithms have only considered point objects, though in many cases we have to handle sets of extended objects such as rectangles. In this paper we present an approach for non-point clustering. The main idea is to represent objects (approximated by their minimum bounding rectangles - MBRs) by their vertices. Then a well-defined clustering algorithm can be applied on the set of vertices while a refinement step follows to identify the final clusters of objects. We compare the performance of our approach with the naive solution of representing objects by their MBR centers. Our approach results in better partitioning in all studies. We also theoretically show that it will always perform at least as well as the case of considering the MBR centers.

1 Introduction

The management of the huge amount of data is an important requirement by many applications. There are many efforts in the area of data mining, which aim at automating the process of data analysis and discovering interesting patterns from large databases [7, 8]. One of the most important data mining tasks is *clustering*, which aims at discovering groups and interesting distributions in the underlying data so as to be used in knowledge discovery [3, 16].

A number of clustering algorithms have been proposed in the literature [1, 5, 6, 9, 10, 11, 13, 15, 16, 18]. To the best of our knowledge, the majority of them assume multidimensional point objects treating the issue of non-point objects insufficiently [12]. However, in many applications, such as spatio-temporal databases and medical applications, one would prefer searching for compact and well-separated groups of line segments, polygons or volumes.

A generalization of DBSCAN is presented in [19], aiming at the clustering of spatially extended objects. It relies on a density-based notion of clusters and is designed to discover clusters of arbitrary shape. However, the quality of results depends on some user-defined density criteria, i.e., the neighborhood predicate that expresses the notion of neighborhood for the specific application and the cardinality function. Also, an effort in [15] proposes a clustering method, termed *WaveCluster*, which aims at handling spatial databases. It exploits signal-processing techniques to convert the spatial data to

the frequency domain and then finds dense regions in the new space. According to this approach an object may be represented by a feature vector (i.e., a point in n-dimensional space) and then clustering is applied in the feature space (instead of the objects themselves). It detects arbitrary shape clusters but it is not efficient in high dimensional space. Moreover, it is a complicated method based on the usage of the appropriate signal processing techniques so as to represent data objects and find dense regions (clusters) in the underlying data set. Our work is different on the following aspects:

- We exploit the information included in the data objects themselves and there is no need to use any complicated method in order to transform data objects.
- We also propose a method that elaborates on the results of well-known clustering algorithms applied on the features of objects and efficiently identify the "correct" set of objects' clusters.

Moreover, we have applied our approach on real-life spatial data sets. In the sequel we discuss in brief the non-point clustering problem and the main idea of our methodology.

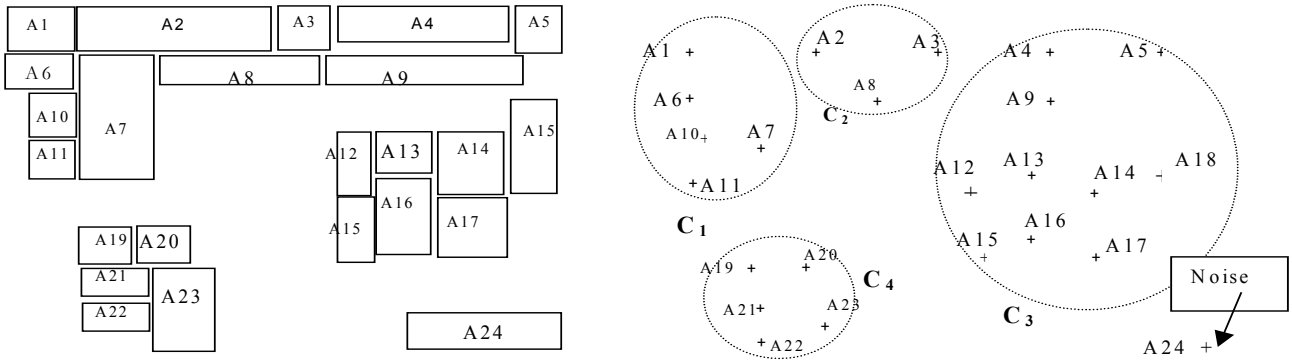
Problem Formulation.

Figure 1a illustrates a set of rectangles (rectangular shapes are popular in the spatial database literature; non-rectangular shapes can be approximated by their minimum bounding (hyper-) rectangles [4, 14]). The goal is to assign these rectangles to a number of clusters. The problem can be formally defined as follows: *Given a data set of n non-point objects, find a partitioning of it into groups (clusters) with respect to some similarity measure or distance metric*. In general terms, the goal is the members of a cluster to have a high degree of similarity, i.e., in geometrical terms to be close to each other and the clusters themselves to be widely spaced.

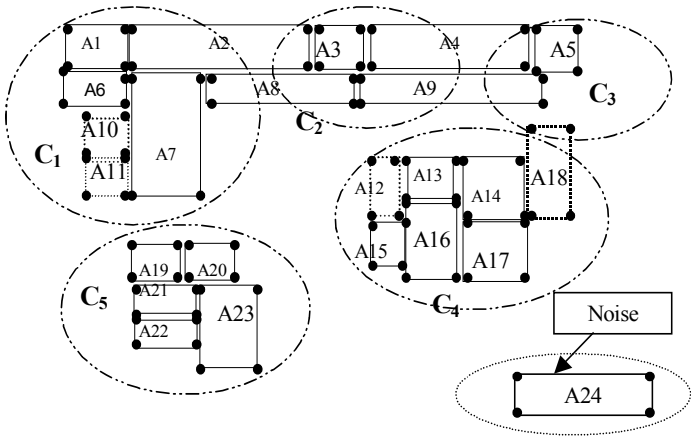
A profound solution in this case is to represent objects by their MBR centers. Then we can apply one of the well-known clustering algorithms proposed in the literature, such as BIRCH [18], CURE [9], DBSCAN [6], etc., on the extracted data set of objects' centers. However, this mapping may ignore useful information about the original data set of objects leading to a clustering scheme that does not fit the data well. It implies that the extracted clusters may not resemble the actual groups in which the data set can be partitioned.

For instance, we assume the data set of non-point objects illustrated in Figure 1a. We extract MBRs' centers and the resulting point data set is presented in Figure 1b. It is obvious from this mapping that there is important information about the data set, regarding the real size of objects and their relative position (the position of an object with respect to others), which is missed. A clustering algorithm applied on this data set would partition it into four clusters as Figure 1b depicts. However, these are not the real clusters in which our data set can be partitioned, as can be easily seen by comparing Figure 1a and Figure 1b. For instance, contrary to what is the case in their centers, rectangle A9 is closer to A8 than to A12 or A13. Thus, the application of a clustering algorithm on the set of rectangles' centers results on grouping A9 with A12, A13 in C_3 and A8 with A2, A3 in C_2. It is obvious that there exist many cases in which the previous described solution does not work well.

In order to address this problem we propose a new approach for Non-Point objects Clustering, called NPClu, that is based on representing objects by their MBRs' vertices.



(a)Groups of MBRs (b) Clustering of MBRs' centers



(c) Clustering of MBRs' vertices

Figure 1. Different approaches for clustering

The rest of the paper is organized as follows. In Section 2 we present the main steps of NPClu. Section 3 analyzes the fundamental step of the proposed method (i.e., refinement step), while it discusses integrity issues related to non-point clustering methodology. In Section 4, we present the experimental evaluation of NPClu in comparison with the naïve approach using synthetic and real data sets. This is followed by an analysis of NPClu complexity. Finally, in Section 5 we offer concluding remarks and we indicate directions for further research.

2 The NPClu Algorithm

Our approach for clustering non-point data is based on three distinct steps, a *preprocessing step*, *a clustering step* and a *refinement step*.

- In the *preprocessing step*, the data set of objects (e.g. rectangles) is represented in a d-dimensional space by their MBRs' vertices. Thus, a d-dimensional object is represented by 2^d points in d-dimensional space. A set of points is defined that corresponds to the MBRs' vertices of initial set of objects, called as *transformed data set*. Since we handle large spatial data sets, a R*-tree is built based on the transformed data set[1].

[1] We use the R*-tree since it is generally accepted as one of the most efficient R-tree variants.

- Then, in the *clustering step*, a well-established clustering algorithm for points is applied in order to discover significant groups of vertices in the transformed data set. This procedure results in a set of clusters where the vertices of a (hyper-) rectangle may be assigned to either no cluster, or a single cluster, or more than one clusters.
- As a consequence, there is a need for a *refinement step*. It elaborates on the initial clustering results and, using some distance criteria, defines the final partitioning of the original set. More specifically, clusters may be merged and/or vertices may be moved from one cluster to another so as the vertices of a (hyper-) rectangle to be classified into the same cluster. Furthermore, there are cases that some (hyper-) rectangles can be considered as outliers.

Before describing in detail the above methodology, we introduce some notions that play significant role in the context of the clustering phase. We note that in the sequel, where we use the term rectangle we mean a hyper-rectangle.

Definition 1. A rectangle R is termed as *resolved* if all of its vertices are classified into the same cluster c_i. The vertices of P are called *resolved vertices* for the c_i.

Definition 2. A rectangle R is termed as *unresolved* if its vertices are classified into different clusters.

Definition 3. Let x, y two vertices of rectangle R. If the vertex x is classified into cluster c_i and y into c_j then x is termed as *an unresolved vertex of cluster* c_j and y *unresolved vertex of cluster* c_i.

In the sequel, we describe each step in detail.

2.1 Preprocessing step

The algorithm starts with the preprocessing phase in which the basic structures, used in clustering phase, are built. It includes two steps:

- *Mapping*: Let S_{obj} a set of (non-point) objects. A mapping of objects to their MBR vertices results in a transformed data set of rectangles, S_{vert}. Each object is represented by the 2^d vertices of its MBR, thus we have the set of MBRs' vertices $S_{vert} = \{(P_i(x_1,\ldots, x_d), P_i)| P_i \in S_{obj}\}$ where $P_i(x_1,\ldots, x_d)$ is a vertex of the MBR(P_i). Obviously, $|S_{vert}| = 2^d \cdot |S_{obj}|$.
- *Building a R*-tree*: We build a R*-tree [2] for the set of rectangles' vertices, further called $R(S_{vert})$. This is used by the following clustering step in order to find the nearest neighbors of a rectangle. Since $R(S_{vert})$ is an index of points, its nodes at the leaf level consist of entries of the following structure: *entry* = {*id*, *coordinates*, *cluster_id*}, where *id* = {*rect_id*, *vertex_id*}, $0 \le vertex\_id \le 2^d-1$, *coordinates* = $(x_1,\ldots, x_d)$ and *cluster_id* is a value to be filled later (in the clustering step that follows).

2.2 Clustering step

Once the set of vertices and the corresponding R*-tree have been built up, we proceed with the clustering phase. Actually, we apply a clustering algorithm to S_{vert}, which discovers clusters of arbitrary shape in underlying data and handles efficiently the outliers (DBSCAN is a good example of such an algorithm). Once clustering has been completed, we are aware of the clusters into which each of the rectangles' vertices $P_i(x_1,\ldots, x_d)$ is classified as well as of the vertices defined as outliers/noise. Based

on this information, $R(S_{vert})$ is also properly updated, i.e., *cluster_id* value is filled with the appropriate value.

Clearly, after the clustering step we have only an indication of the objects' classification since the corresponding vertices may have been classified into different clusters, some or all of them may have been defined as noise, etc. Thus, a refinement step is necessary so as to handle the different cases of this problem and define the final partitioning of the underlying set of rectangles.

2.3 Refinement step

The main tasks of this step are as follows:

1. For each cluster i defined in the clustering step, we find:
   i. the rectangles whose all vertices are classified to the same cluster, so as to define the set of *resolved* rectangles of the clusters, Res_rect_{ci}. (For instance, rectangle A5 is a resolved rectangle of cluster C_3 as Figure 1c depicts),
   ii. the rectangles whose vertices are classified to more than one clusters. The vertices of these rectangles define the set of *unresolved* rectangles of the cluster, Unr_rect_{ci}. (In Figure 1c rectangle A18 is considered as unresolved for clusters C_3 and C_4).
   iii. the vertices of rectangles that considered as noise/outlier define the set of noise for c_i (rectangle A24 is consider as noise in the set presented in Figure 1c).
2. For the set of defined clusters $C=\{c_i, i=1, \ldots, numc\}$, we calculate the average intra-cluster distance of clusters based on Res_rect_{ci}. Also the cluster size of each of the clusters c_i is defined as the maximum distance[2] between two rectangles classified into c_i. In case that there are no resolved rectangles in a cluster, we consider the MBR of the vertices that belong to this cluster. In this case only, both the intra-cluster distance and the size of the cluster are defined as to be the longer edge of this MBR.

3 Finding the objects' clusters

Based on the above definitions and some distance criteria we proceed to define the final partitioning of the rectangles' set. The clusters are merged or rectangles are moved from one cluster to another so that rectangles whose vertices split into different clusters to be considered into the same cluster. Also, the outliers can be discovered and handled efficiently.

More specifically, we consider the following cases of the problem regarding the classification of a rectangle:

Case 1. All the vertices of a rectangle are defined as noise. In this case the rectangle is also considered as noise.

Case 2. Only one cluster is involved in clustering results. We may consider the following two sub-cases.

   i. all the rectangle vertices are classified in a cluster. In this case, the rectangle is considered as *resolved* and it belongs to the cluster of their vertices.

[2] The distance between two rectangles is defined as the minimum distance between two rectangles' vertices.

ii. at least one of the rectangle vertices is defined as noise and the rest belong to a cluster, c_i. We compare the cluster size with the length of the largest unresolved rectangle edge, *rect_length,* and we decide how to handle the rectangle. More specifically, if the *cluster_size* is larger than *rect_length* the rectangle is assigned to c_i, otherwise we consider the rectangle as noise.

Case 3. More than one clusters are involved in clustering results. The rectangle vertices are classified in more than one clusters (2, 3, …, 2^d clusters). In the cases that the number of involved clusters, denoted *cl_inv*, is less than 2^d some of the vertices may be considered as noise. In order to decide where the rectangle (further referred as unresolved rectangle) can finally be classified we are based on some distance criteria. More specifically, if the maximum *cluster_size* among the clusters involved in clustering results is smaller than *rect_length,* the unresolved rectangle is considered as noise. In the other case, we compare the distance of the unresolved rectangle from its nearest neighbor, in each of the involved clusters, with the maximum intra-cluster distance of the involved clusters. Based on this distance, we decide to merge clusters or assign the unresolved rectangle to one of the involved clusters. The basic step of this decision can be summarized as follows:

If max(cluster_size) < rect_length
Rectangle is noise
else if rectangle is near more than one clusters
merge these clusters
else
assign rectangle to its nearest cluster.

Here we define when a rectangle is considered to be "near" a cluster.

Definition 4. Let P_i an unresolved rectangle of clusters c_i and c_j. We also assume its nearest neighbors *neigh_c_i* and *neigh_c_j* in the clusters c_i and c_j respectively. Then, P_i is considered as to be near both c_i and c_j if $d_{ci} < \max(\text{intra\_}d_{ci}, \text{intra\_}d_{cj})$ and $d_{cj} < \max(\text{intra\_}d_{ci}, \text{intra\_}d_{cj})$, where $\text{intra\_}d_{ci}$, $\text{intra\_}d_{cj}$ is the intra-cluster distance of c_i and c_j respectively while d_{ci}, d_{cj} is the distance of P_i from its nearest neighbor in cluster c_i and c_j respectively.

In Figure 2, the refinement step of the NPClu algorithm is sketched. We note that the order in which we examine the unresolved rectangles is based on some criteria so as to efficiently result in the final partitioning. Our approach starts with the clusters containing the highest number of unresolved rectangles so as to result more effectively in a small number of involved clusters. Then, we use the *Sort_desc* procedure to sort the unresolved rectangles of the considered cluster. *Sort_desc* sorts rectangles in a descending order with regard to the number of different clusters into which the vertices of rectangles are classified. Also, *Select_rect* procedure helps us to select the rectangle that is classified in clusters with the highest number of resolved rectangles. In general terms, these procedures enable the discovery and efficient handling of rectangles considered as noise/outliers.

```
NPClu_refinement (SP, RSvert)
{ clustering_alg(Svert)
For each cluster ci
   For each rectangle Pj
   {
      If  Pj(k)(x,y).cluster_id == ci , ∀ k=1,…2^d.
         Add Pj(k)(x,y) into Res_rect_ci
      else if Pj(m)(x, y).cluster_id ≠ ci  and
         ∃ Pj(k)(x, y).cluster_id == ci , where  m≠k
            Add Pj (m)(x, y) to Unr_rect_ci
         else if Pj(k)(x,y).cluster_id == ci , ∀ k=1,…, 2^d.
            Add Pj (m)(x, y) to Noise
         else
            j=j+1 //next rectangle
      If  Res_rect_ci. ≠  ∅
         {
            dci = average intra-cluster distance in ci,
            cluster_size_ci= max{dist(Pi,Pj)|Pi,Pj∈ci, i≠j}
         }
         else
         {
            MBRci = MBR of the vertices classified into ci
            intra_dci = edge of MBRci ,
            cluster_size_ci = edge of MBRci
      }
      while (∃ rectangles classified in more than one clusters)
      {
         Find the cluster that shares the minimum number of rectangles with
         other clusters. Let ci.
         while  Unr_rect_ci. ≠  ∅
         {
            Sort_desc( Unr_rect_ci.)
            Select_rect(Unr_rect_ci)
            If (cl_inv ==1) {
               If (cluster_sizeci < rect_length(Pk))
                     add Pk into Noise
                  else
                     assign Pk to ci
               }
            else  // cl_inv >1
            {
               neigh_ci =  nearest neighbor of P(k  in ci,,
               neigh_cj = nearest neighbor of P(k in cj,j=1,…,cl_inv
               dci = dist (Pk, neigh_ci)
               dcj= dist (Pk, neigh_cj) , ∀ j=1,…,cl_inv
               if (max_j=1,…cl_inv(cluster_sizeci, cluster_sizecj) < rect_length(Pk))
                     Add Pk into Noise
               else
               {
                  For  j=1 to cl_inv
                     if(dci<max(inta_dci,intra_dcj),and
                     dcj<max(inta_dci,intra_dcj)),∀j=1,…,cl_inv
                        ci = ci U cj    //merge the clusters
                     else
                     {
                        if dci< dcj
                           assign Pk  to resolved clusters of ci
                        else if dci> dcj
                           assign Pk  to resolved clusters of cj
                        else
                           assign Pk to to resolved clusters of the cluster
                           with max inta-cluster distance
                     }
               }
               Update the number of clusters num_c, and
               redefine the set of resolved and unresolved clusters.
            }
      }
   }
}
```

Figure 2. Sketch of the refinement step of the NPClu algorithm

The refinement procedure is iterative. It is repeated until there are no unresolved rectangles, that is, the algorithm terminates when all rectangles of the underlying set have been classified or have been defined as noise. The implementation of our approach is based on DBSCAN. It is a well-known density-based algorithm that combines our requirements for clustering: i) discovery of clusters with arbitrary shape and ii) efficiency on large databases. Nevertheless, we may consider any other clustering algorithm in order to discover significant groups in the set of rectangles' vertices.

3.1 An example

As an example, we assume a data set of rectangles (Figure 1a) and we apply our clustering approach.

Step 1: The mapping of rectangles to their vertices is presented in Figure 1c.

Step 2: A clustering algorithm for points would partition the transformed data set to the clusters described by the cycles in Figure 1c. Thus the results of clustering would be described as follows:

Clusters	Noise
c_1 = {A1, A2, A6, A7, A8, A10, A11} c_2 = {A2, A3, A4, A8, A9} c_3 = {A4, A5, A9, A18} c_4 = {A12, A13, A14, A15, A16, A17, A18} c_5 = {A19, A20, A21, A22, A23}	{A24}

Step 3: The information about the unresolved and resolved rectangles for the discovered clusters is as follows:

Information about unresolved and resolved rectangles	
Unresolved_c_1 = {((A2, A8), c_2)}	Resolved_c_1 = {A1, A6, A10, A11}
Unresolved_c_2 = {((A2, A8), c_1), ((A4, A9),c_3)}	Resolved_c_2 = {A3}
	Resolved_c_3 = {A5}
Unresolved_c_3 = {((A4,A9, c_3)), (A18, c_4)}	Resolved_c_4 = {A12, A13, A14, A15, A16, A17}
Unresolved_c_4 = {(A18,c4)}	Resolved_c_5 = {A19, A20, A21, A22, A23}
Unresolved_c_5 = $\varnothing$	

The cluster that shares the highest number of rectangles with other clusters is c_2. More specifically, it shares rectangles both with c_1 and c_3, i.e., $c_1 \cap c_2$ ={A2, A8}, $c_2 \cap c_3$ ={A4, A9}. Since c_1 is the cluster with highest number of resolved rectangle we select one of A2 and A8. Consider the rectangle A2. In this case the number of involved clusters is two (cl_inv=2). Also, we observe that the *rect_length* of A2 is smaller than the maximum cluster size of c_1 and c_2. Then, we proceed to compare the distance of A2 from its nearest neighbors with the intra-cluster distance of c_1 and c_2. The nearest neighbor of A2 is A1 in c_1 and A3 in c_2. Figure 1c depicts that the distance of A2 from both A1 and A3 is smaller than max (intra_dc_1, intra_dc_2} and thus we decide to merge the clusters. The new set of clusters is:

Clusters	Noise
c_{12}={A1, A2, A3, A4, A6, A7, A8, A9, A10, A11} c_3 ={A4, A5, A8, A9, A18} c_4 = {A12, A13, A14, A15, A16, A17, A18} c_5 = {A19, A20, A21, A22, A23}	{A24}

while a cluster that shares rectangles with more than one clusters is c_3.

Unresolved_c_3 = {(A9,c_{12}), (A4,c_{12}), (A18,c_4)}.

The cluster c_{12} contains the highest number of resolved rectangles, thus we select to examine one of the rectangles A4 or A9. Assume the rectangle A4. We observe that *rect_length* of A4 is smaller than max(cluster_sizec$_{12}$, cluster_sizec$_3$). Also the distance of A4 from its nearest neighbors in c_{12} and c_3 (A3 and A5 respectively) is less than max (intra_dc$_{12}$, intra_dc$_3$). Therefore we decide to merge the two involved clusters c_{12} and c_3. As a result, we produce the cluster c_{123} with P_{c123} ={A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11}.

Then we make decision about the classification of A18, which splits its vertices in c_{123} and c_4. Based on the above-described distance criteria we decide that A18 is nearest to c_4 than c_{123} and thus we assign it to c_4. Finally, the clustering process would result to the following clustering:

Clusters	Noise
c_{123} = {A1, A2, A3, A4, A5, A6, A7, A8, A9, A10, A11} c_4 = {A12, A13, A14, A15, A16, A17, A18} c_5 = {A19, A20, A21, A22, A23}	{A24}

Recalling the result achieved by considering the centers of rectangles (Figure 1b), it is obvious that the proposed clustering approach leads to better results.

3.2 Integrity issues

One would argue that the NPClu methodology, especially the refinement step, needs integrity checking so that this step does not violate the integrity of the results provided by the clustering step. In the sequel we analyze how NPClu treats different cases of the initial clustering results so as to define the final partitioning of an objects' set. In the following lemmas we summarize these cases giving also their respective proof sketches.

Lemma 1: *If NPClu clustering step discovers the inherent partitioning of the data set, the refinement step does not change the partitioning.*

Proof sketch: In this case the clustering algorithm partitions the data set of rectangles into the correct number clusters, that is all the vertices of a rectangle belong into the same cluster. Since there are no unresolved rectangles the refinement step is not applied. The output of complete algorithm is identical to the clustering step.

Lemma 2: *If NPClu clustering step discovers more clusters than the inherent number, then the refinement step discovers the correct partitioning of the data set of rectangles.*

Proof sketch: Assume that the clustering step partitions the transformed data set of rectangles (data set of vertices) into more than the actual number of clusters that appear in the data set of rectangles. Then, there are rectangles that are shared between clusters and the refinement step is applied. The clusters are merged or rectangles are moved from one cluster to another so that the set of clusters in which the data set of rectangles can be partitioned is defined.

4 NPClu Evaluation

In this section we present an experimental study of the proposed methodology using data sets of 2D rectangles. We compare it with the naive approach that considers the centers of MBRs in order to define the data set on which clustering is applied and identify the clusters in the set of rectangles. The implementation of the proposed algorithm described in Section 3 is written in C++. In the experiments we used DBSCAN [6] at the first clustering step of NPClu, and the implementation of R*-tree as presented in [2]. We chose DBSCAN due to its acceptable complexity and also due to its ability to detect clusters of skewed geometry. Nevertheless, any clustering algorithm can be used at this step to partition the set of vertices. NPClu uses the clustering algorithm only to define the initial set of clusters on which the main step of defining clusters of objects (i.e., the refinement step) is based.

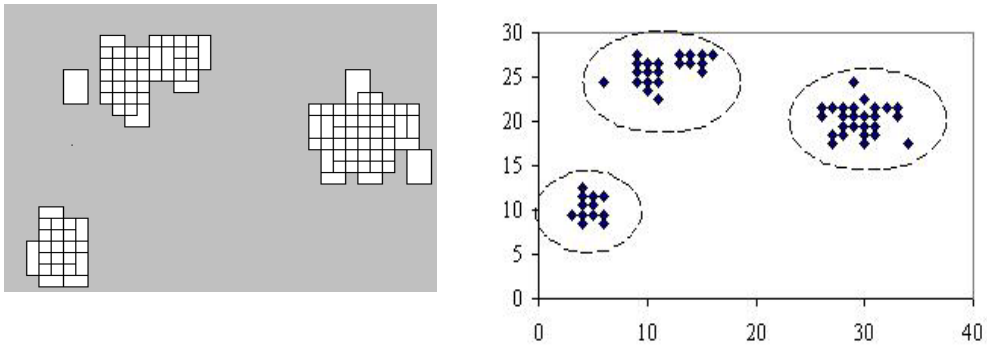
4.1 Experimental evaluation

We used synthetic and real data sets in order to evaluate the performance of the proposed methodology for non-point objects clustering. More specifically, we used the following types of data sets:

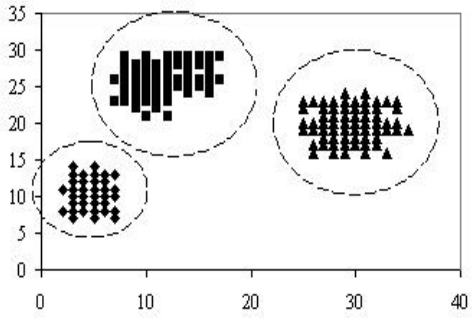
- Three synthetic data sets, which represent different cases of the refinement step application. These cases are: i) refinement step is not applied (all rectangles are resolved), ii) merging of clusters, iii) moving of objects between clusters.
- Two synthetic sets of rectangles with arbitrary-shaped clusters. They were based on a synthetic dataset that is presented in [6]. This data set was used to verify NPClu behavior in the case of arbitrary shaped clusters.
- Two real data sets. The first set represents a part of German railways while the second set represents the towns and villages of some Greek islands. Both data sets, available in [17], are good for the purpose of testing NPClu since naïve clustering fails in discovering the actual clusters.

4.1.1 Synthetic data sets

We assume a set of rectangles as presented in Figure 3a. It is obvious that there are three clusters of rectangles in this data set. The mapping of rectangles to their centers is presented in Figure 3b. The application of DBSCAN on the set of rectangles centers results in a partitioning of the data set depicted by the dotted cycles in Figure 3b. The clusters correspond to the "actual" clusters presented in the set of rectangles. Then we map rectangles (Figure 3a) to their vertices and apply our approach. Figure 3c depicts the results of the clustering on the set of vertices that is a partitioning into three clusters. It is obvious that there are no unresolved rectangles and thus the refinement step is not performed. As a



(a) the data set (b) naïve clustering



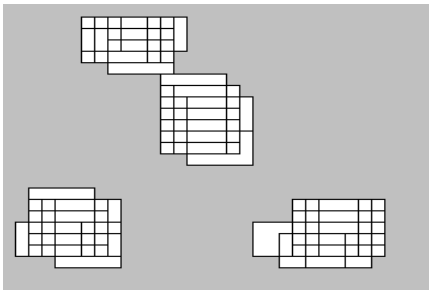
(c) NPClu clustering

Figure 3.The first experiment: the refinement step of NPClu is not required

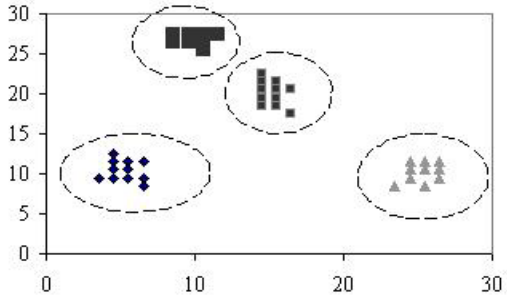
consequence, our approach gives the same results as in the case where we consider the centers of rectangles.

The following two experiments show that our approach gives better results than the naïve approach. Figure 4a presents a data set of rectangles, which, according to the clustering criteria introduced in previous sections, can be partitioned into 3 clusters. The mapping of rectangles to their centers is presented in Figure 4b. We apply DBSCAN on this data set and the resulting partitioning is a set of four clusters as shown by the dotted cycles in Figure 4b. It is clear that the clustering approach based on centers of rectangles does not work properly in this case. Then we consider the set of rectangles vertices (Figure 4c) and we apply DBSCAN on this data set so as to identify significant clusters in the underlying set of vertices. Comparing Figure 4a and Figure 4c we can detect unresolved rectangles, i.e., vertices shared between two or more clusters. Thus, the refinement step is applied in order define the final clusters depicted in Figure 4d. It is obvious that NPClu identifies the correct clusters on which the data set of rectangles can be partitioned by merging the clusters of Figure 4c.

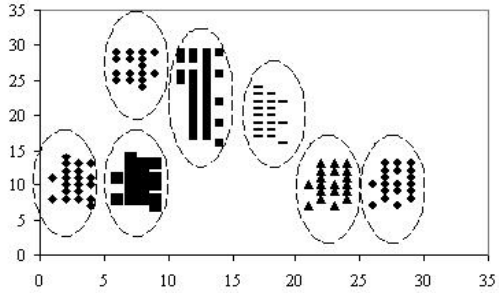
A similar experiment is presented in Figure 5. Figure 5a depicts the set of rectangles representing by their centers as well as how this set can be partitioned by DBSCAN. On the other hand, Figure 6b shows the vertices of the rectangles set while Figure 5d depicts the final partitioning of data set as defined by our approach. In this case the final partitioning is defined (Figure 5d) if we consider the initial partitioning of DBSCAN as presented in Figure 5c and then we move unresolved rectangles (i.e., R1, R2 and R3 in Figure 5a) from one cluster to another during the refinement step.



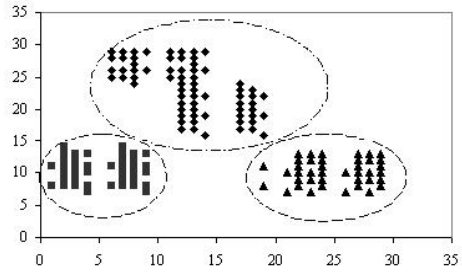
(a) the data set



(b) naïve clustering

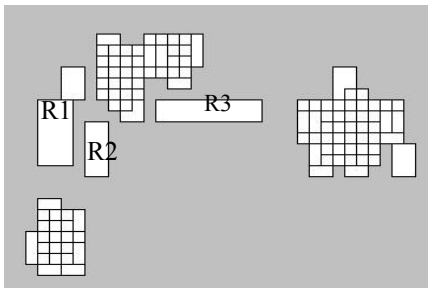


(c) NPClu clustering – before the refinem ent step takes place

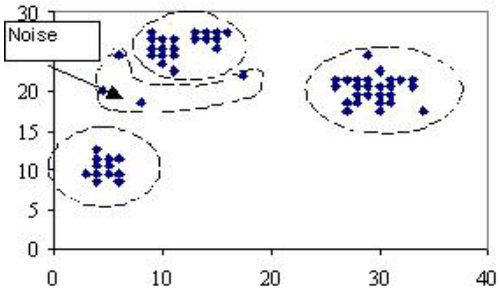


(d) NPClu clustering – after the refinement step takes place

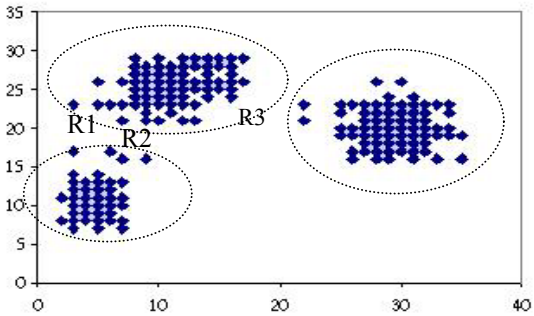
Figure 4. The second experiment: the refinement step of NPClu is required



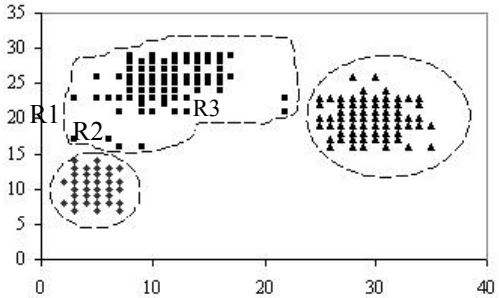
(a) the data set



(b) naïve clustering



(c) NPClu clustering – before the refinement step takes place



(d) NPClu clustering – after the refinement step takes place

Figure 5. The third experiment: the refinement step of NPClu is required

The following experiments show that NPClu works well in the case of *arbitrary shaped* clusters of non-point spatial objects. Moreover, we prove that our approach works as well as the approach based on centers when we consider a data set of small rectangles almost zero-point sized. In case that the data set consists of larger rectangles moreover prolonged in one of their dimensions, our approach results in better partitioning since the rectangles' centers cannot successfully represent them. Figure 6 depicts a dataset of points used in the following study as centers for defining a set of rectangles. The cycles in Figure 6 depict the partitioning of the data set as defined by DBSCAN.

Assuming the set of points in Figure 6 as rectangles' centers, we define a set of rectangles depicted in Figure 7a. The partitioning as defined by non-point clustering approach is presented in Figure 7b. It is obvious that the proposed clustering approach discovers the correct number of clusters as good as the approach based on the rectangles' centers.

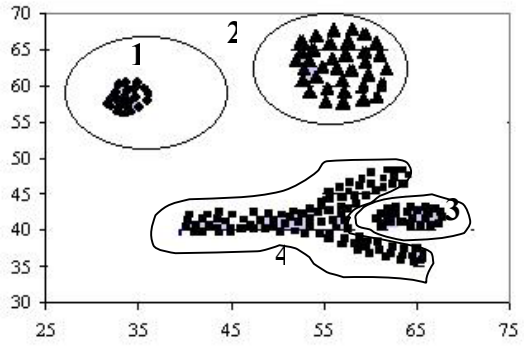
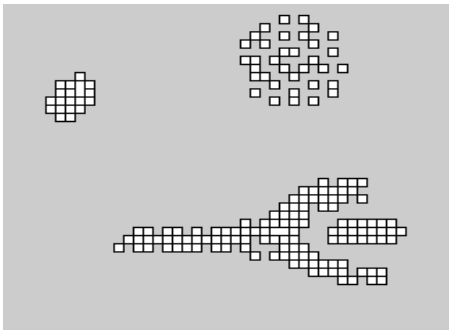
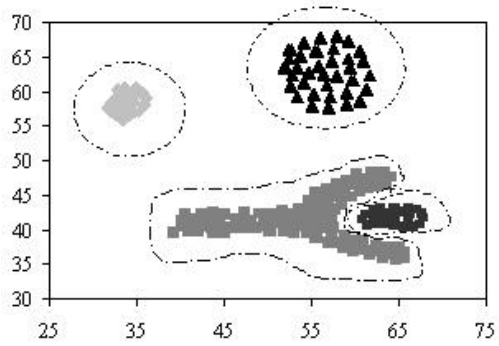


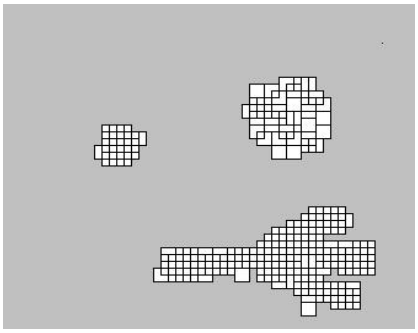
Figure 6. A set of point used as centers for defining the set of rectangles in Figure 7a and Figure 8a.

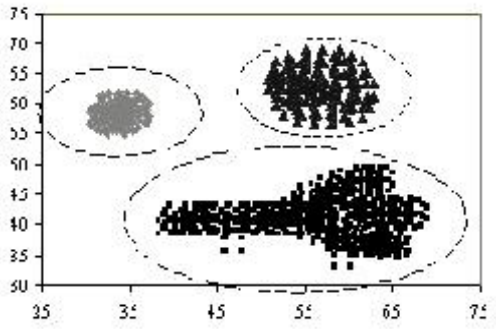




(a) the data set (b) NPClu clustering – after the refinement step takes place

Figure 7. The fourth experiment: A set of rectangles based on the data set of Figure 6 as centers and edge = 0.2





(a) the data set (b) NPClu clustering – after the refinement step takes place

Figure 8 The fifth experiment:A set of rectangles based on the data set of Figure 7 as centers and edge = 2

Then we define a set of rectangles based on the same centers as the above set but with larger edge (Figure 8a). It is clear that the inherent number of clusters is equal to 3. However, if we consider the approach that is based on the rectangles' centers the clustering results will be the same as defined on the previous experiment (i.e., 4 clusters). On the other hand, we consider the set of the rectangle vertices (Figure 8b) and we apply the proposed clustering approach. It is clear that the actual clusters (i.e., 3) are extracted for the considered dataset.

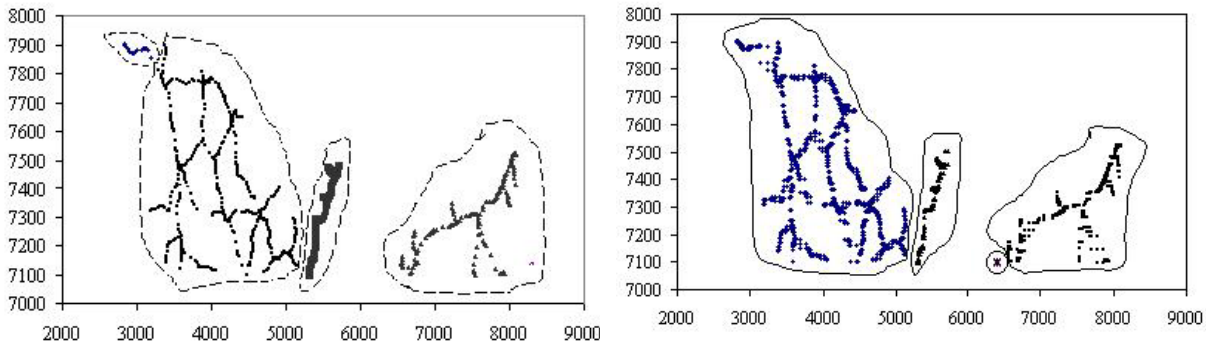
4.1.2 Real data sets

We also evaluated the proposed clustering approach using real data sets and we found that in each case NPClu results in better partitioning than the naïve approach. More specifically, we consider the set of rectangles representing a part of the German railways [17]. The set of the rectangles' centers and the extracted clusters as defined by DBSCAN is presented in Figure 9a. Then, we apply NPClu approach to the same data set, which defines the set of clusters depicted in Figure 9b.

A similar experiment was carried out assuming a set of rectangles representing the towns and villages of some Greek islands [17]. The clustering results as produced by the naive approach and the NPClu approach are depicted in Figure 10a and Figure 10b, respectively. We observe that an island or a group of islands is ignored (labeled as noise) when we apply clustering to the centers of rectangles as Figure 10a shows. On the contrary, NPClu identifies all the significant groups of islands cities in underlying data. Thus, it is clear that important knowledge can be ignored or not exploited in the clustering approach based on centers of rectangles.

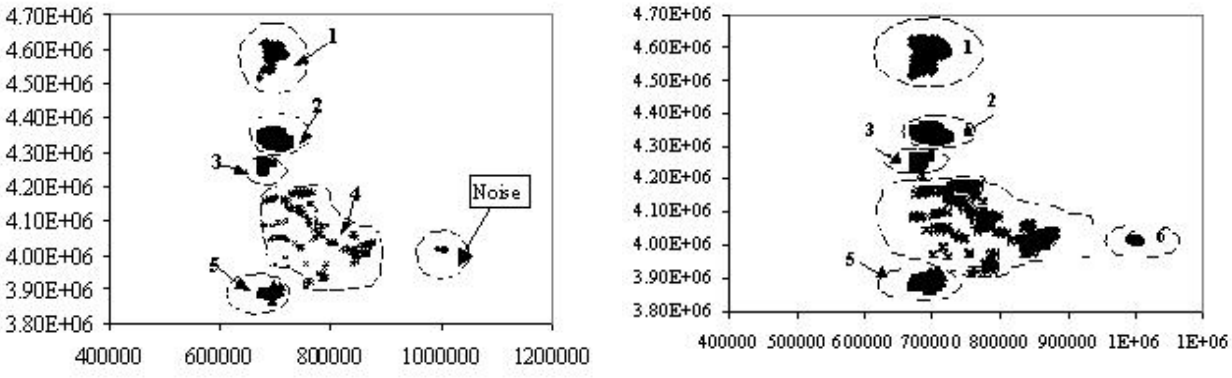
4.2 Complexity issues

The complexity of our approach is based on the complexity of the three steps described in Section 2, that is, the preprocessing, the clustering and the refinement step. The complexity of the first step is related to the mapping of objects to their MBR vertices and construction of the R*-tree, complexity O(n), where n the number of rectangles. The second step, which aims at discovering significant groups in the set of vertices, depends on the complexity of considered clustering algorithm. For example the complexity of DBSCAN is O(n·logn). The major step of our methodology is the refinement step, which also results in the definition of the final partitioning. In the sequel, we present in more detail the complexity of this step.



(a) Clustering based on centers (naïve clustering) (b) NPClu clustering

Figure 9 The sixth experiment:German raillines



(a) Clustering based on centers (naïve clustering) (b) NPClu clustering

Figure 10. The seventh experiment:Greek islands

The refinement step considers unresolved rectangles to find the final partitioning of the rectangles set. Based on the initial clustering results we define for each cluster the resolved and unresolved clusters, the complexity of this process is $O(c \cdot n^2)$, where c is the number of clusters and n is the number of rectangles. We assume that *unr_cl* is the number of clusters containing unresolved rectangles, *unr_rect* is the number of unresolved rectangles in the whole data set while res_rect_c is the number of resolved rectangles in a cluster containing unresolved rectangles. Then, the complexity of the process regarding the decision of merging clusters or assigning rectangles to a specific cluster in order to achieve the final partitioning is,

$$O(unr\_cl \cdot (unr\_rect \cdot \log n + res\_rect_c^2)).$$

This step is applied only to clusters with unresolved rectangles whose number decreases significantly during the refinement step. Therefore, we can assume that res_rect_c is much smaller than n. Also, usually *unr_cl* and *unr_rect* << n. Then, assuming that we have defined the set of resolved and unresolved rectangles, the complexity of finding the final set of objects' clusters is compatible with O(logn). On the other hand, in case that res_rect_c is the number of rectangles the complexity of the refinement step will be $O(n^2)$. However, as we have already mentioned this is not a usual case.

Based on the above discussion we conclude that the overall complexity of NPClu is $O(n^2)$.

To quantify this, we experimented with data sets containing different percentages of unresolved rectangles and we estimated the time complexity of the refinement step in comparison to the clustering step complexity. Figure 11 depicts the ratio of NPClu refinement step time to the clustering step time (when we use DBSCAN) as function of the percentage of unresolved rectangles in the data set. We use the set of rectangles presented in the evaluation study of NPClu (see Section 4.1) as well as two additional synthetic datasets with the appropriate number of unresolved rectangles for our experiments. We note, here, that the time complexity of NPClu does not only depend on the number of unresolved rectangles but also on the iterations of the refinement step so as to conclude to the final partitioning. For instance, in case of 15.5% unresolved rectangles the time needed to result in final partitioning is higher than in case of 24% as in the second case we find the final partitioning only by merging two of the clusters.

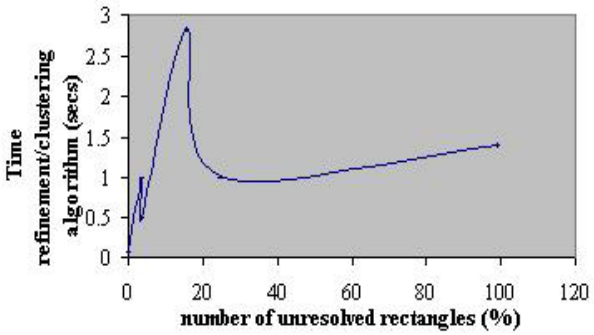


Figure 11. Time complexity of the refinement step vs clustering step w.r.t. portion of unresolved rectangles

In general terms, it is clear from the above discussion that the complexity of the non-point clustering methodology is comparable to the cost of the clustering step.

5 Conclusions

Clustering is an important task in managing voluminous data so as to identify significant groups in an underlying data set and extract "interesting" knowledge from it. Since it is widely recognized as a major tool in a number of business or scientific applications, several clustering techniques and algorithms have been proposed and are available in the literature. The vast majority of algorithms have only considered point objects, though in many cases we have to handle sets of extended objects such as (hyper)-rectangles. In this paper we presented an algorithm (NPClu) for clustering non-point objects (i.e., objects that have some extent rather that being points). To the best of our knowledge, this is an open research issue since the related work is very limited. NPClu consists of three steps.

In the first step of NPClu, objects (approximated by their minimum bounding rectangles - MBRs) are represented by their vertices. In the second step, a clustering algorithm is applied on the set of vertices while at the third refinement-stage the final clusters of objects are identified.

We compared the performance of NPClu to the naive solution of representing objects by their MBR centers. Our approach results in better partitioning in all cases. We have theoretically shown that it will always perform at least as well as the naïve case. The experimental evaluation also shows that the computational cost of the refinement step in NPClu is comparable to the cost of the clustering step.

Further work will be devoted towards the following directions:

- Applying NPClu in related application domains such as medical or cadastre contexts, where groups of polygons or areas can be identified.
- Addressing scaling issues, where the relationship of the proportion of the non-resolved rectangles be related to the efficiency of the algorithm.
- Studying the efficiency of NPClu in dimensionality d>2.

Acknowledgements

We thank C. Amanatidis for his help in the initial stages of the implementation. We are also grateful to Dr Joerg Sander for providing information and the source code for DBSCAN.

References

[1] R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan, “Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications”, *Proceedings of ACM SIGMOD Conference*, 1998.

[2] N. Beckmann, H.-P. Kriegel, R. Scheider, B. Seeger, “The R*-tree: an Efficeinet and Robust Access Method for Points and Rectangles”, *Proceedings of ACM SIGMOD Conference,* 1990.

[3] M. J. A. Berry, G. Linoff. *Data Mining Techniques For marketing, Sales and Customer Support*. John Willey & Sons, Inc, 1996.

[4] T. Brinkhoff, H.-P. Kriegel, B. Seeger, “Efficient Processing of Spatial Joins using R-trees”, Proceedings of ACM SIGMOD Conference , 1993.

[5] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, X. Xu. "Incremental Clustering for Mining in a Data Warehousing Environment*", Proceedings of 24th VLDB Conference,* New York, USA, 1998.

[6] M. Ester, H.-P. Kriegel, J. Sander, X. Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*", Proceedings of 2nd Int. Conf. On Knowledge Discovery and Data Mining,* Portland, OR, 1996.

[7] U. M. Fayyad, G. Piatesky-Shapiro, P. Smuth, R. Uthurusamy. *Advances in Knowledge Discovery and Data Mining*. AAAI Press 1996

[8] U. Fayyad, R. Uthurusamy. "Data Mining and Knowledge Discovery in Databases", *Communications of the ACM.* Vol.39, No11, November 1996.

[9] S. Guha, R. Rastogi, K. Shim, "CURE: An Efficient Clustering Algorithm for Large Databases", *Proceedings of the ACM SIGMOD Conference*, 1998.

[10] S. Guha, R. Rastogi, K. Shim, "ROCK: A Robust Clustering Algorithm for Categorical Attributes", *Proceedings of the IEEE Conference on Data Engineering*, 1999.

[11] A. Hinneburg, D. Keim. "An Efficient Approach to Clustering in Large Multimedia Databases with Noise". *Proceedings of KDD Conference,* 1998.

[12] A.K Jain, M.N. Murty, P.J. Flyn. “Data Clustering: A Review”, *ACM Computing Surveys*, Vol. 31, No3, September 1999.

[13] R. Ng, J. Han. "Efficient and Effective Clustering Methods for Spatial Data Mining". *Proceedings of the 20th VLDB Conference,* 1994.

[14] J. Orenstein, “Spatial Query Processing in an Object-Oriented Database System”, *Proceedings of ACM SIGMOD International Conference*, 1986.

[15] C. Sheikholeslami, S. Chatterjee, A. Zhang. "WaveCluster: A-MultiResolution Clustering Approach for Very Large Spatial Database", *Proceedings of 24th VLDB Conference, New York*, USA, 1998.

[16] S. Theodoridis, K. Koutroubas. *Pattern Recognition,* Academic Press, 1999

[17] Y. Theodoridis. Spatial Datasets: an "unofficial" collection. Available at http://dias.cti.gr/~ytheod/research/datasets/spatial.html

[18] T. Zhang, R. Ramakrishnman, M. Linvy. "BIRCH: An Efficient Method for Very Large Databases", *Proceedings of ACM SIGMOD Conference,* Montreal, Canada, 1996.

[19] J. Sander, M. Ester, H.-P. Kriegel, X. Xu. "A Density-Based Algorithm in Spatial Databases: The Algorithm DBSCAN and Its Applications*", Data Mining and Knowledge Discovery,* pp.169-194, 1998.