Available online at www.sciencedirect.com

ScienceDirect

Data & Knowledge Engineering 60 (2007) 208–221

DATA & KNOWLEDGE ENGINEERING

www.elsevier.com/locate/datak

ST-DBSCAN: An algorithm for clustering spatial–temporal data

Derya Birant *, Alp Kut

Department of Computer Engineering, Dokuz Eylul University, 35100 Izmir, Turkey

Available online 13 March 2006

Abstract

This paper presents a new density-based clustering algorithm, ST-DBSCAN, which is based on DBSCAN. We propose three marginal extensions to DBSCAN related with the identification of (i) core objects, (ii) noise objects, and (iii) adjacent clusters. In contrast to the existing density-based clustering algorithms, our algorithm has the ability of discovering clusters according to non-spatial, spatial and temporal values of the objects. In this paper, we also present a spatial–temporal data warehouse system designed for storing and clustering a wide range of spatial–temporal data. We show an implementation of our algorithm by using this data warehouse and present the data mining results.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Data mining; Cluster analysis; Spatial–temporal data; Cluster visualization; Algorithms

1. Introduction

Clustering is one of the major data mining methods for knowledge discovery in large databases. It is the process of grouping large data sets according to their *similarity*. Cluster analysis is a major tool in many areas of engineering and scientific applications including data segmentation, discretization of continuous attributes, data reduction, outlier detection, noise filtering, pattern recognition and image processing. In the field of Knowledge Discovery in Databases (KDD), cluster analysis is known as unsupervised learning process, since there is no priori knowledge about the data set.

Most studies in KDD [12] focus on discovering clusters from ordinary data (non-spatial and non-temporal data), so they are impractical to use for clustering spatial–temporal data. Spatial–temporal data refers to data which is stored as temporal slices of the spatial dataset. Knowledge discovery from spatial–temporal data is a very promising subfield of data mining because increasingly large volumes of spatial–temporal data are collected and need to be analyzed. The knowledge discovery process for spatial–temporal data is more complex than for non-spatial and non-temporal data. Because spatial–temporal clustering algorithms have to consider the spatial and temporal neighbors of objects in order to extract useful knowledge. Clustering algorithms

* Corresponding author. Tel./fax: +90 232 3736040.
E-mail address: derya@cs.deu.edu.tr (D. Birant).

0169-023X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2006.01.013

designed for spatial–temporal data can be used in many applications such as geographic information systems, medical imaging, and weather forecasting.

This paper presents a new density-based clustering algorithm ST-DBSCAN, which is based on the algorithm DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) [5]. In DBSCAN, the density associated with a point is obtained by counting the number of points in a region of specified radius around the point. Points with a density above a specified threshold are constructed as clusters. Among the existing clustering algorithms, we have chosen DBSCAN algorithm, because it has the ability in discovering clusters with arbitrary shape such as linear, concave, oval, etc. Furthermore, in contrast to some clustering algorithms, it does not require the predetermination of the number of clusters. DBSCAN has been proven in its ability of processing very large databases [3,5,6]. We have improved DBSCAN algorithm in three important directions. First, unlike the existing density-based clustering algorithms, our algorithm can cluster spatial–temporal data according to its non-spatial, spatial and temporal attributes. Second, DBSCAN cannot detect some noise points when clusters of different densities exist. Our algorithm solves this problem by assigning to each cluster a *density factor*. Third, the values of border objects in a cluster may be very different than the values of border objects in opposite side, if the non-spatial values of neighbor objects have little differences and the clusters are adjacent to each other. Our algorithm solves this problem by comparing the average value of a cluster with new coming value.

In addition to new clustering algorithm, this paper also presents a spatial data warehouse system designed for storing and clustering a wide range of spatial–temporal data. Environmental data, from a variety of sources, were integrated as coverages, grids, shapefiles, and tables. Special functions were developed for data integration, data conversion, visualization, analysis and management. User-friendly interfaces were also developed allowing relatively inexperienced users to operate the system. In order demonstrate the applicability of our algorithm to real world problems, we applied our algorithm to the data warehouse, and then presented and discussed the data mining results.

Spatial–temporal data is indexed and retrieved according to spatial and time dimensions. A time period attached to the spatial data expresses when it was valid or stored in the database. A temporal database may support valid time, transaction time or both. Valid time denotes the time period during which a fact is true with respect to the real world. Transaction time is the time period during which a fact is stored in the database. This study focuses on valid time aspect of temporal data.

The rest of the paper is organized as follows. Section 2 summaries the existing clustering algorithms and gives basic concepts of density-based clustering algorithms. Section 3 describes the drawbacks of existing density-based clustering algorithms and our efforts to overcome these problems. Section 4 explains our algorithm in detail and presents the performance of the algorithm. Section 5 presents three applications which are implemented to demonstrate the applicability of it to real world problems. It shows and discusses the data mining results. Finally, a conclusion and some directions for future works are given in Section 6.

2. Related works and basic concepts

This section summaries and discusses the existing clustering algorithms and then gives basic concepts of density-based algorithms.

2.1. Density-based clustering

The problem of clustering can be defined as follows:

Definition 1. Given a database of n data objects $D = \{o_1, o_2, \ldots, o_n\}$. The process of partitioning D into $C = \{C_1, C_2, \ldots, C_k\}$ based on a certain similarity measure is called clustering, C_i's are called clusters, where $C_i \subseteq D$, $(i = 1, 2, \ldots, k)$, $\bigcap_{i=1}^{k} C_i = \emptyset$ and $\bigcup_{i=1}^{k} C_i = D$.

Clustering algorithms can be categorized into five main types [13]: *Partitional*, *Hierarchical*, *Grid-based*, *Model-based* and *Density-based* clustering algorithms. In *Partitional* algorithms, cluster similarity is measured in regard to the mean value of the objects in a cluster, center of gravity, (*K*-Means [19]) or each cluster is represented by one of the objects of the cluster located near its center (*K*-Medoid [26]). *K* is an input parameter for these algorithms, unfortunately it is not available for many applications. CLARANS [20] is an improved version of *K*-Medoid algorithm for mining in spatial databases. *Hierarchical* algorithms such as CURE [9], BIRCH [31] produce a set of nested clusters organized as a hierarchical tree. Each node of the tree represents a cluster of a database *D*. *Grid-based* algorithms such as STING [28], WaveCluster [23] are based on multiple-level grid structure on which all operations for clustering are performed. In *Model-based* algorithms (COBWEB [8], etc.), a model is hypothesized for each of the clusters and the idea is to find the best fit of that model to each other. They are often based on the assumption that the data are generated by a mixture of underlying probability distributions.

The *Density-based* notion is a common approach for clustering. Density-based clustering algorithms are based on the idea that objects which form a *dense* region should be grouped together into one cluster. They use a fixed threshold value to determine *dense* regions. They search for regions of high density in a feature space that are separated by regions of lower density.

Density-based clustering algorithms such as DBSCAN [5], OPTICS [2], DENCLUE [15], CURD [18] are to some extent capable of clustering databases [21]. One drawback of these algorithms is that they capture only certain kinds of noise points when clusters of different densities exist. Furthermore, they are adequate if the clusters are distant from each other, but not satisfactory when clusters are adjacent to each other. The detailed description of these problems and our solutions are given in Section 3.

In our study, we have chosen DBSCAN algorithm, because it has the ability in discovering clusters with arbitrary shape such as linear, concave, oval, etc. Furthermore, in contrast to some clustering algorithms, it does not require the predetermination of the number of clusters. DBSCAN has been proven in its ability of processing very large databases [3,6].

In the literature, DBSCAN algorithm was used in many studies. For example, the other popular density-based algorithm OPTICS (Ordering Points To Identify the Clustering Structure) [2] is based on the concepts of DBSCAN algorithm and identifies nested clusters and the structure of clusters. Incremental DBSCAN [7] algorithm is also based on the clustering algorithm DBSCAN and is used for incremental updates of a clustering after insertion of a new object to the database and deletion of an existing object from the database. Based on the formal notion of clusters, the incremental algorithm yields the same result as the non-incremental DBSCAN algorithm. SDBDC (Scalable Density-Based Distributed Clustering) [16] method also uses DBSCAN algorithm on both local sites and global site to cluster distributed objects. In this method, DBSCAN algorithm is firstly carried out on each local site. Then, based on these local clustering results, cluster representatives are determined. Then, based on these local representatives, the standard DBSCAN algorithm is carried out on the global site to construct the distributed clustering. This study proposes the usage of different Eps-values for each local representative. Wen et al. [29] adopted DBSCAN and Incremental DBSCAN as the core algorithms of their query clustering tool. They used DBSCAN to cluster frequently asked questions and most popular topics on a search engine. Spieth et al. [24] applied DBSCAN to identify solutions for the inference of regulatory networks. Finally, SNN density-based clustering algorithm [25] is also based on DBSCAN and it is applicable to high-dimensional data consisting of time series data of atmospheric pressure at various points on the earth.

2.2. Basic concepts

DBSCAN is designed to discover arbitrary-shaped clusters in any database *D* and at the same time can distinguish noise points. More specifically, DBSCAN accepts a radius value Eps(ε) based on a user defined distance measure and a value MinPts for the number of minimal points that should occur within Eps radius. Some concepts and terms to explain the DBSCAN algorithm can be defined as follows [5].

Definition 2 (*Neighborhood*). It is determined by a distance function (e.g., Manhattan Distance, Euclidean Distance) for two points *p* and *q*, denoted by $\mathrm{dist}(p,q)$.

Definition 3 (*Eps-neighborhood*). The Eps-neighborhood of a point *p* is defined by $\{q \in D \mid \mathrm{dist}(p,q) \leqslant \mathrm{Eps}\}$.

Definition 4 (*Core object*). A core object refers to such point that its neighborhood of a given radius (Eps) has to contain at least a minimum number (MinPts) of other points (Fig. 1c).

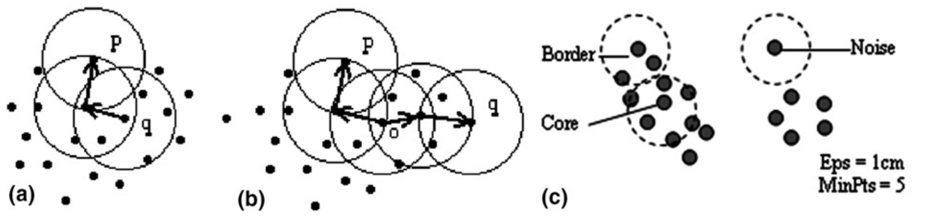

Fig. 1. Basic concepts and terms: (a) p density-reachable from q, (b) p and q density-connected to each other by o and (c) border object, core object and noise.

Definition 5 (*Directly density-reachable*). An object p is directly density-reachable from the object q if p is within the Eps-neighborhood of q, and q is a core object.

Definition 6 (*Density-reachable*). An object p is density-reachable from the object q with respect to Eps and MinPts if there is a chain of objects $p_1, \ldots, p_n$, $p_1 = q$ and $p_n = q$ such that p_{i+1} is directly density-reachable from p_i with respect to Eps and MinPts, for $1 \leqslant i \leqslant n$, $p_i \in D$ (Fig. 1a).

Definition 7 (*Density-connected*). An object p is density-connected to object q with respect to Eps and MinPts if there is an object $o \in D$ such that both p and q are density-reachable from o with respect to Eps and MinPts (Fig. 1b).

Definition 8 (*Density-based cluster*). A cluster C is a non-empty subset of D satisfying the following "maximality" and "connectivity" requirements:

(1) $\forall p,q$: if $q \in C$ and p is density-reachable from q with respect to Eps and MinPts, then $p \in C$.
(2) $\forall p,q \in C$: p is density-connected to q with respect to Eps and MinPts.

Definition 9 (*Border object*). An object p is a border object if it is not a core object but density-reachable from another core object.

The algorithm starts with the first point p in database D, and retrieves all neighbors of point p within Eps distance. If the total number of these neighbors is greater than MinPts—if p is a core object—a new cluster is created. The point p and its neighbors are assigned into this new cluster. Then, it iteratively collects the neighbors within Eps distance from the core points. The process is repeated until all of the points have been processed.

3. Problems of existing approaches

3.1. Problem of clustering spatial–temporal data

In order to determine whether a set of points is *similar* enough to be considered a cluster or not, we need a distance measure dist(i,j) that tells how far points i and j are. The most common distance measures used are Manhattan distance, Euclidean distance, and Minkowski distance. Euclidean distance is defined as Eq. (1).

$$\mathrm{dist}(i,j) = \sqrt{(x_{i1}-x_{j1})^2 + (x_{i2}-x_{j2})^2 + \cdots + (x_{in}-x_{jn})^2} \tag{1}$$

where $i = (x_{i1}, x_{i2}, \ldots, x_{in})$ and $j = (x_{j1}, x_{j2}, \ldots, x_{jn})$ are two n-dimensional data objects. For example, the Euclidean distance between the two data objects $A(1,2)$ and $B(5,3)$ is 4.12.

DBSCAN algorithm uses only one distance parameter Eps to measure *similarity* of spatial data with one dimension. In order to support two dimensional spatial data, we propose two distance metrics, Eps1 and Eps2, to define the *similarity* by a conjunction of two density tests. Eps1 is used for *spatial values* to measure the closeness of two points geographically. Eps2 is used to measure the *similarity* of *non-spatial values*. For example, $A(x1,y1)$ and $B(x2,y2)$ are two points (spatial values), $t1$, $t2$ (*DayTimeTemperature, NightTime Temperature*) and $t3$, $t4$ are four temperature values of these points respectively (non-spatial values). In this example, Eps1 is used to measure the closeness of two points geographically, while Eps2 is used to measure the *similarity* of temperature values. If $A(x1,y1,t1,t2)$ and $B(x2,y2,t3,t4)$ are two points, Eps1 and Eps2 are calculated by the formulas in Eq. (2).

$$\begin{aligned}\text{Eps1} &= \sqrt{(x1-x2)^2+(y1-y2)^2}\\ \text{Eps2} &= \sqrt{(t1-t2)^2+(t1-t2)^2}\end{aligned} \tag{2}$$

In order to support temporal aspects, spatio-temporal data is first filtered by retaining only the temporal neighbors and their corresponding spatial values. Two objects are temporal neighbors if the values of these objects are observed in consecutive time units such as consecutive days in the same year or in the same day in consecutive years.

3.2. Problem of identifying noise objects

From the view of a clustering algorithm, noise is a set of objects not located in clusters of a database. More formally, noise can be defined as follows:

Definition 10 (*Noise*). Let $C_1,\ldots,C_k$ be the clusters of the database D. Then the *noise* is the set of points in the database D not belonging to any cluster C_i, where $i=1,\ldots,k$, i.e., noise $=\{p\in D\,|\,\forall i\colon p\notin C_i\}$.

Existing density-based clustering algorithms [14] produce meaningful and adequate results under certain conditions, but their results are not satisfactory when clusters of different densities exist. To illustrate, consider the example given in Fig. 2. This is a simple dataset containing 52 objects. There are 25 objects in the first cluster C_1, 25 objects in the second cluster C_2, and two additional noise objects o_1 and o_2. In this example, C_2 forms a denser cluster than C_1. In other words, the densities of the clusters are different from each other. DBSCAN algorithm identifies only one noise object o_1. Because approximately for every object p in C_1, the distance between the object p and its nearest neighbor is greater than distance between o_2 and C_2. For this reason, we can't determine an appropriate value for the input parameter Eps. If the Eps value is less than the distance between o_2 and C_2, some objects in C_1 are assigned as noise object. If the Eps value is greater than the distance between o_2 and C_2, the object o_2 is not assigned as noise object.

The example in Fig. 2 shows that DBSCAN algorithm is not satisfactory when clusters of different densities exist. In order to overcome this problem, we propose a new concept: *density factor*. We assign to each cluster a *density factor*, which is the degree of the density of the cluster. We begin with the notion of *density_distance*.

Definition 11 (*Density_distance*). Let *density_distance_max* of an object p denote the maximum distance between the object p and its neighbor objects within the radius Eps. Similarly, let *density_distance_min* of an object p denote the minimum distance between the object p and its neighbor objects within the radius Eps.

(i) density_distance_max $(p)=\max\{\text{dist}(p,q)\,|\,q\in D\wedge \text{dist}(p,q)\leqslant \text{Eps}\}$,
(ii) density_distance_min$(p)=\min\{\text{dist}(p,q)\,|\,q\in D\wedge \text{dist}(p,q)\leqslant \text{Eps}\}$.

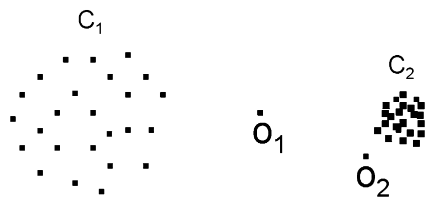

Fig. 2. Example data set which contains clusters with different densities.

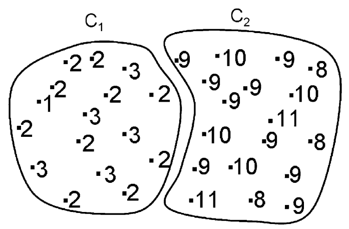

Fig. 3. Example data set which contains adjacent clusters.

The *density_distance* of an object *p* is defined as density_distance_max(*p*)/density_distance_min(*p*).

So far, we define *density factor* of a cluster as follows.

Definition 12 (*Density_factor*). The *density_factor* of a cluster *C* is defined as Eq. (3).

$$\text{density\_factor}(C) = 1 \Bigg/ \left[\frac{\sum_{p \in C} \text{density\_distance}(p)}{|C|} \right] \tag{3}$$

The *density factor* of a cluster *C* captures the degree of the density of the cluster. If *C* is a "loose" cluster, density_distance_min would increase and so the *density distance* in Definition 11 would be quite small, thus forcing the *density factor* of *C* to be quite close to 1. Otherwise, if *C* is a "tight" cluster, density_distance_min would decrease and so the *density distance* in Definition 11 would be quite big, thus forcing the *density factor* of *C* to be quite close to 0.

3.3. Problem of identifying adjacent clusters

The existing density-based clustering algorithms [17] are adequate if the clusters are distant from each other, but not satisfactory when clusters are adjacent to each other (Fig. 3). If the values of neighbor objects have little differences, the values of border objects in a cluster may be very different than the values of other border objects in opposite side. In other words, the value of a border object may be very different than the value of the most far border object. Because small value changes on neighbors can cause big value changes between starting points and ending points of a cluster. However cluster objects should be within a certain distance from the cluster means. We solve this problem by comparing the average value of a cluster with new coming value. If the absolute difference between Cluster_Avg( ) and Object_Value is bigger than the threshold value, $\Delta\epsilon$, then the new object is not appended to the cluster. Cluster_Avg( ) refers to the average or mean value of the objects contained in the cluster. Object_Value refers to the non-spatial value of the object such as temperature value of a location.

4. ST-DBSCAN algorithm

4.1. The description of the algorithm

While DBSCAN algorithm needs two inputs, our algorithm ST-DBSCAN requires four parameters Eps1, Eps2, MinPts, and $\Delta\epsilon$ because of the extensions described in Section 3. Eps1 is the distance parameter for spatial attributes (latitude and longitude). Eps2 is the distance parameter for non-spatial attributes. A distance metric such as Euclidean, Manhattan or Minkowski Distance Metric can be used for Eps1 and Eps2. MinPts is the minimum number of points within Eps1 and Eps2 distance of a point. If a region is dense, then it should contain more points than MinPts value. In [5], a simple heuristic is presented which is effective in many cases to determine the parameters Eps and MinPts. The heuristic suggests MinPts $\approx \ln(n)$ where *n* is the size of the database and Eps must be picked depending on the value of MinPts. The first step of the heuristic method is to determine the distances to the *k-nearest neighbors* for each object, where *k* is equal to MinPts. Then these *k-distance values* should be sorted in descending order. Then we should determine the threshold point which is the first "valley" of the sorted graph. We should select Eps to less than the distance defined by the first valley. The last parameter $\Delta\epsilon$ is used to prevent the discovering of combined clusters because of the little differences in non-spatial values of the neighboring locations.

The algorithm starts with the first point *p* in database *D* and retrieves all points density-reachable from *p* with respect to Eps1 and Eps2. If *p* is a core object (see Definition 4), a cluster is formed. If *p* is a border object (see Definition 9), no points are density-reachable from *p* and the algorithm visits the next point of the database. The process is repeated until all of the points have been processed.

As shown in Fig. 4, the algorithm starts with the first point in database *D* (i). After processing this point, it selects the next point in *D*. If the selected object does not belong to any cluster (ii), *Retrieve_Neighbors* function is called (iii). A call of *Retrieve_Neighbors*(*object*, Eps1, Eps2) returns the objects that have a distance less than Eps1 and Eps2 parameters to the selected object. In other words, *Retrieve_Neighbors* function retrieves all objects *density-reachable* (see Definition 6) from the selected object with respect to Eps1, Eps2, and MinPts. The result set forms the Eps-Neighborhood (see Definition 3) of the selected object. *Retrieve_Neighbours*(*object*, Eps1, Eps2) equals to the intersection of *Retrieve_Neighbours*(*object*, Eps1) and *Retrieve_Neighbours*(*object*, Eps2). If the total number of returned points in Eps-Neighborhood is smaller than MinPts

```
Algorithm ST_DBSCAN (D, Eps1, Eps2, MinPts, Δε)
  // Inputs:
      // D={o1, o2, ..., on}  Set of objects
      // Eps1 : Maximum geographical coordinate (spatial) distance value.
      // Eps2 : Maximum non-spatial distance value.
      // MinPts : Minimum number of points within Eps1 and Eps2 distance.
      // Δε : Threshold value to be included in a cluster.
  // Output:
      // C={C1, C2, … Ck} Set of clusters

 Cluster_Label = 0

 For i=1 to n                                  //(i)
   If oi is not in a cluster Then              //(ii)
     X=Retrieve_Neighbors(oi , Eps1, Eps2)     //(iii)

     If |X| < MinPts Then
         Mark oi as noise                      //(iv)
     Else                        //construct a new cluster (v)
        Cluster_Label = Cluster_Label + 1

        For j=1 to |X|
          Mark all objects in X with current Cluster_Label
        End For

        Push(all objects in X)                 //(vi)

        While not IsEpmty()
           CurrentObj = Pop()
           Y= Retrieve_Neighbors(CurrentObj, Eps1, Eps2)

           If |Y| >= MinPts Then
              ForAll objects o in Y            //(vii)
                 If (o is not marked as noise or it is not in a cluster) and
                    |Cluster_Avg() - o.Value| <= Δε Then
                    Mark o with current Cluster_Label
                    Push(o)
                 End If
              End For
           End If
        End While
     End If
   End If
 End For
End Algorithm
```

Fig. 4. ST_DBSCAN algorithm.

input, the object is assigned as noise (iv). This means that the selected point has not enough neighbors to be clustered. The points which have been marked to be noise may be changed later, if they are not directly density-reachable (see Definition 5) but they are density-reachable (see Definition 6) from some other point of the database. This happens for border points of a cluster.

If the selected point has enough neighbors within Eps1 and Eps2 distances—if it is a core object—then a new cluster is constructed (v). Then all directly density-reachable neighbors of this core object are also marked as new cluster label. Then the algorithm iteratively collects density-reachable objects from this core object by using a stack (vi). The stack is necessary to find density-reachable objects from directly density-reachable objects. If the object is not marked as noise or it is not in a cluster, and the difference between the average value of the cluster and the new coming value is smaller than $\Delta\epsilon$, it is placed into the current cluster (vii). After processing the selected point, the algorithm selects the next point in D and algorithm continues iteratively until all of the points have been processed.

When the algorithm searches the neighbors of any object by using *Retrieve_Neighbors* function (line (iii) in the algorithm), it takes into consideration both spatial and temporal neighborhoods. The non-spatial value of an object such as a temperature value is compared with the non-spatial values of spatial neighbors and also with the values of temporal neighbors (previous day in the same year, next day in the same year, and the same day in other years). By this way, non-spatial, spatial and temporal characteristics of data are used in clustering when the algorithm is applied on the table which contains temporal values, beside spatial and non-spatial values.

If two clusters C_1 and C_2 are very close to each other, a point p may belong to both, C_1 and C_2. In this case, the point p must be a border point in both C_1 and C_2. The algorithm assigns point p to the cluster discovered first.

4.2. Performance evaluation

The average runtime complexity of the DBSCAN algorithm is $O(n * \log n)$, where n is the number of objects in the database. Our modifications do not change the runtime complexity of the algorithm. DBSCAN has been proven in its ability of processing very large databases. The paper [6] shows that the runtime of other clustering algorithms such as CLARANS [20], DBCLASD [30] is between 1.5 and 3 times the runtime of DBSCAN. This factor increases with increasing size of the database.

As in all databases, fast access to raw data in spatial–temporal databases depends on the structural organization of the stored information and the availability of suitable indexing methods. While a well designed data structure can facilitate to rapidly extract the desired information from a set of data, suitable indexing methods can provide to quickly locate single or multiple objects [1]. Well known spatial indexing techniques include Quadtrees [22], *R*-Trees [11], *X*-Tree [4] and others, see [10] for an overview. An *R-Tree* is a spatial indexing technique that stores information about spatial objects such as object ids, the Minimum Bounding Rectangles (MBR) of the objects or groups of the objects. Each entry of a leaf node is of the form (R, P) where R is a rectangle that encloses all the objects that can be reached by following the node pointer P. In our study, we made an improvement of the *R*-Tree indexing method to handle spatial–temporal information. We created some nodes in *R*-Tree for each spatial object and linked them in temporal order. During the application of the algorithm, this tree is traversed to find the spatial or temporal neighbor objects of any object. Two objects are temporal neighbors if the values of these objects are observed in consecutive time units such as consecutive days in the same year or in the same day in consecutive years.

In addition to spatial index structure, some filters should also be used to reduce the search space for spatial data mining algorithms. These filters allow the operations on neighborhood paths by reducing the number of paths actually created. They are necessary to speed up the processing of queries.

5. Application

In order to demonstrate the applicability of our algorithm to real world problems; we present three data mining applications by using a spatial–temporal data warehouse. The task of clustering is to discover the regions that have *similar* seawater characteristics. The aim of the clustering is obtain a number of clues about

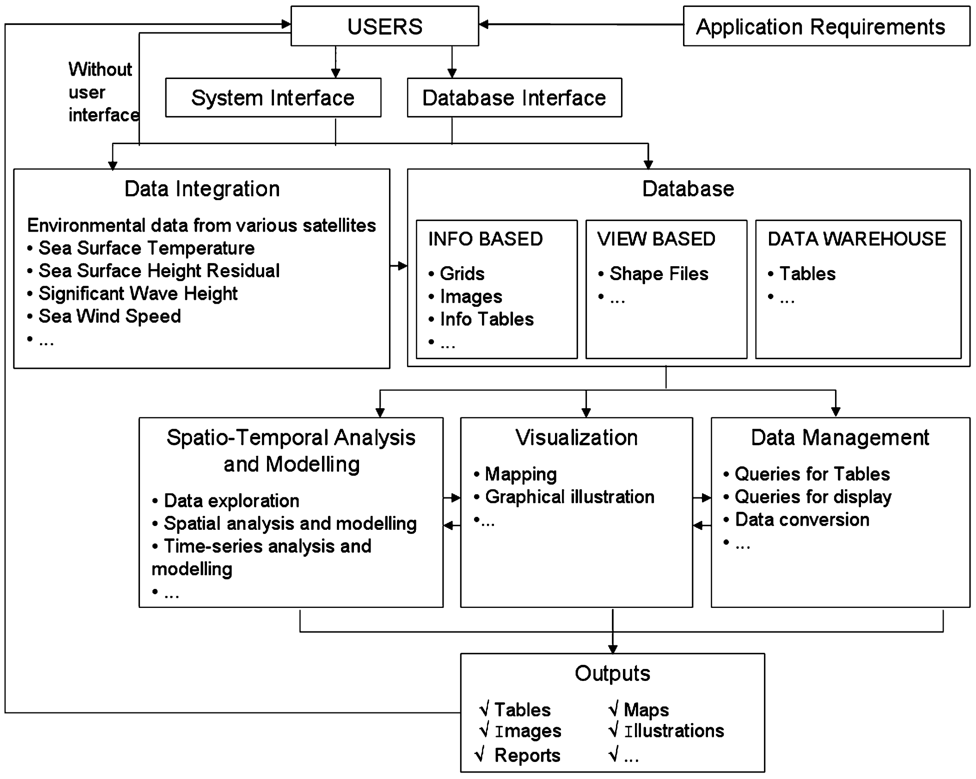

Fig. 5. The schematic diagram of the system.

how the physical properties of the water are distributed in a marine environment. The first application discovers the regions that have *similar* sea surface temperature values. In the second application, the goal is to identify spatially based partitions which have the *similar* sea surface height residual values. The third application includes cluster analysis on significant wave height data.

Fig. 5 shows the structure of the system. In visualization part of the study, remotely sensed data on the historical extent of marine areas were used in a spatial metrics analysis of geographical form of countries and islands. User-friendly interfaces were developed allowing relatively inexperienced users to operate the system. Special functions were developed for data integration, data conversion, query, visualization, analysis and management. Marine environmental data (e.g., sea surface temperature, wave height values, bathymetric data), from a variety of sources, were integrated as coverages, grids, shapefiles, and tables.

The process of KDD involves several steps such as data integration and selection, data preprocessing and transformation, data mining, and the evaluation of the data mining results. Our efforts on each step are described below.

5.1. Data integration and selection

We designed a spatial data warehouse system which contains information about four seas: the Black Sea, the Marmara Sea, the Aegean Sea, and the east of the Mediterranean Sea. These seas surround the countries Turkey to the north, west, south; Greece to the east, south, west; and Cyprus. The geographical coordinates of our work area are 30° to 47.5° north latitude and 17.0° to 42.5° east longitude.

As shown in Fig. 6, the data model contains a central fact table, STATIONS, which interconnects the tables: Sea_Surface_Temperature, Sea_Surface_Height, Wave_Height, and Sea_Winds. The data size is approximately 0.8 GB. In data warehouse, the dimensions are time and space. The time dimension can be grouped into year, month, and day. Similarly, the space dimension can be grouped into StationID RegionID

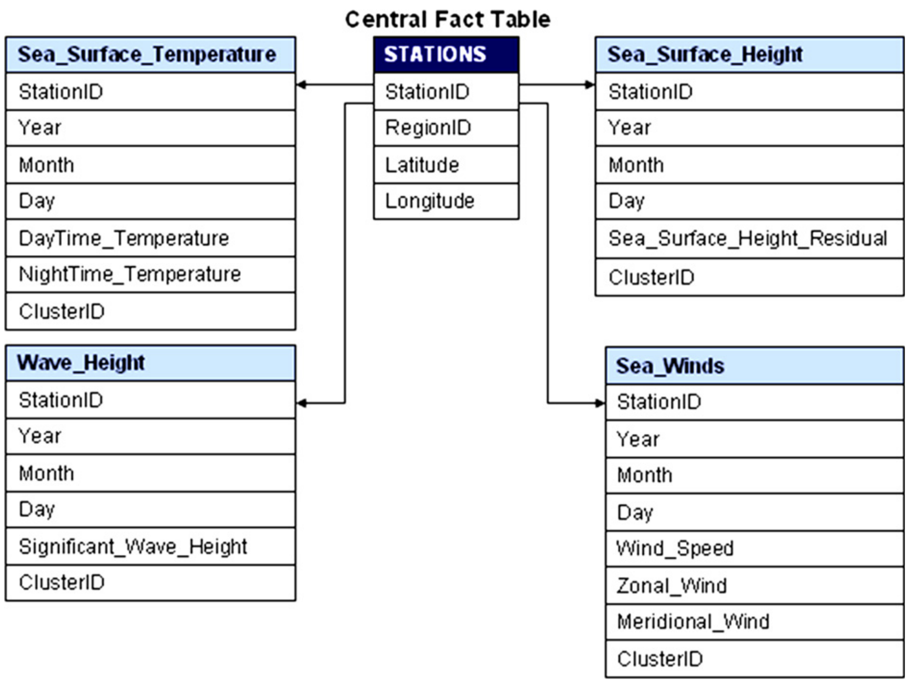

Fig. 6. General overview of data warehouse.

and ClusterID. The first column, StationID, identifies the geographic location of monitoring station. The column, RegionID, identifies the name of the sea (Black Sea, Marmara Sea, Aegean Sea, or Mediterranean Sea). The last column, ClusterID, identifies a particular cluster of stations that have similar characteristics.

Sea_Surface_Temperature table contains weekly daytime and nighttime temperature records, which were measured in years between 2001 and 2004. The data has been provided by NOAA-Series Satellites (National Oceanic and Atmospheric Administration).[1] It contains approximately 1.5 million rows. Data in Sea_Surface_Height table has been provided by Topex/Poseidon Satellite[2] and it was collected over five-day periods in years between 1992 and 2002. Wave_Height table contains significant wave height values which are collected over ten-day periods in years between 1992 and 2002. Similar to the significant sea surface height values, the significant wave height values have also been provided by Topex/Poseidon Satellite. Sea_Winds table contains information about wind speed, zonal wind and meridional wind. The data measured daily in years between 1999 and 2004 has been provided by QuikSCAT Satellite.[3]

5.2. Data preprocessing and transformation

Satellite data generally contain false information and sometimes several values can be missing. We filled missing values with the average of adjacent object values. The missing values generally located at the coasts of the Aegean Sea. Because the Aegean coast is extremely indented with numerous gulfs and inlets.

The maps derived from the NOAA-AVHRR (Polar Orbiting Advanced Very High Resolution Radiometer) are used to compute Sea Surface Temperatures (SSTs) by applying the Multi-Channel Sea Surface Temperature algorithm (MCSST). The latest version of this algorithm uses the following formula in the calculation of the SST:

$$\mathrm{SST} = a * T4 + b * (T4 - T5) * Tf + c * (\sec(q) - 1) * (T4 - T5) - d \tag{4}$$

[1] NOAA/AVHRR Satellite Data Web Site, http://podaac.jpl.nasa.gov/.
[2] Topex/Poseidon Sea Level Grids Description, http://podaac.jpl.nasa.gov/woce/woce3_topex/topex/docs/topex_doc.htm.
[3] QuikSCAT SeaWinds, Gridded Ocean Wind Vectors, http://podaac.jpl.nasa.gov/products/product109.html.

where q is the satellite zenith angle or the incidence angle of the incoming radiation based on the horizontal plane of the satellite, $T4$ and $T5$ are the brightness temperatures from AVHRR channels 4 and 5, respectively. Tf is a first-guess sea surface temperature estimate (obtained from the 1 km MCSST AVHRR mosaic SSTs), and a, b, c, d are empirically derived coefficients [27]. These coefficients are predetermined by comparing AVHRR radiance values to temperature measurements taken from moored and drifting buoys. For example, the nighttime and daytime equations for NOAA-14 Satellite are:

$$\begin{aligned} \text{Daytime SST} &= .9506 * T4 + .0760 * (T4 - T5) * Tf + .6839 * (\sec(0) - 1) * (T4 - T5) - 258.0968 \\ \text{Nighttime SST} &= .9242 * T4 + .0755 * (T4 - T5) * Tf + .6040 * (\sec(0) - 1) * (T4 - T5) - 250.4284 \end{aligned} \tag{5}$$

The Sea Surface Height Residual values are calculated by the formula in Eq. (6).

$$\text{SSHR} = \text{SSH} - \text{MSS} - \text{Tide Effects} - \text{Inverse Barometer} \tag{6}$$

where SSHR is the Sea Surface Height Residual value, SSH is the Sea Surface Height value, and MSS is Mean Sea Surface height value. The residual sea surface is defined as the sea surface height minus the mean sea surface and minus known effects, i.e., tides and inverse barometer.

Significant wave height from TOPEX is calculated from altimeter data based on the shape of a radar pulse after it bounces off the sea surface. A calm sea with low waves returns a sharply defined pulse whereas a rough sea with high waves returns a stretched pulse. The significant wave height is the average height of the highest one-third of all waves in a particular time period.

5.3. Spatial–temporal data mining

In this step of the study, our clustering algorithm is applied three times to discover the spatial–temporal distributions of three physical parameters. The first application uses Sea_Surface_Temperature data to find the regions that have similar sea surface temperature characteristics. The input parameters designated as Eps1 = 3, Eps2 = 0.5, and MinPts = 15. The second application uses Sea_Surface_Height data to find the regions that have similar sea surface height residual values. The input parameters designated as Eps1 = 3, Eps2 = 1, and MinPts = 4. The third application uses Wave_Height table to find the regions that have similar significant wave height values. The input parameters assigned as Eps1 = 1, Eps2 = 0.25, and MinPts = 15. These values for the input parameters are determined by using the heuristics given in [5].

5.4. The evaluation of data mining results

The example database contains weekly daytime and nighttime temperature records, which were measured at 5340 stations in years between 2001 and 2004. In other words, sea surface temperature values stored in the

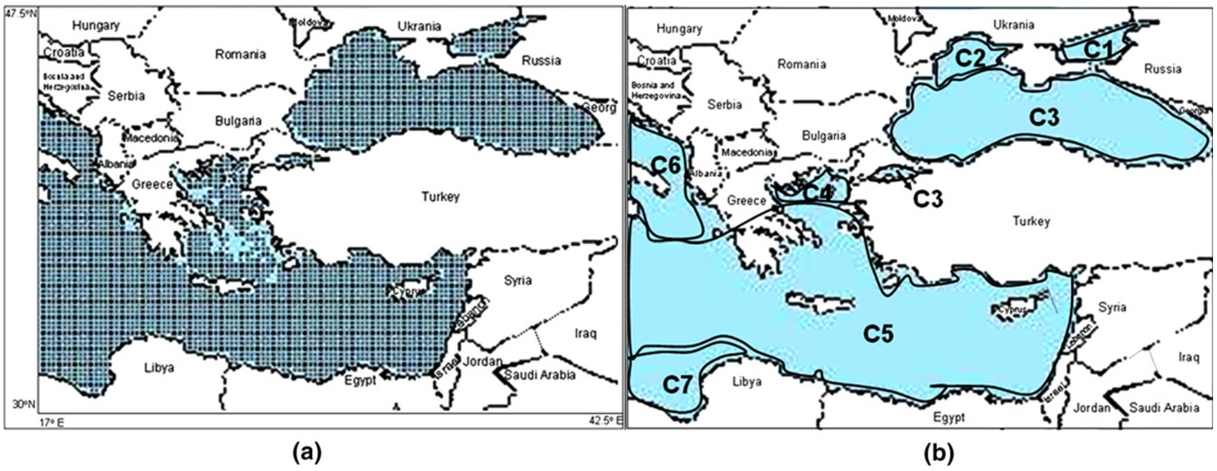

Fig. 7. (a) The locations of 5340 stations. (b) The results of cluster analysis on sea surface temperature data.

database were collected at 5340 stations which are shown in Fig. 7a as black dots. The spatial distribution of temperature in surface water (30–47.5°N and 17–42.5°E) is shown in Fig. 7b. Each cluster has data points that have similar sea surface temperature characteristics. Cluster number 1 is bordered by Ukraine and Russia. This region is the coldest area. Cluster number 2 at the north of the Ukraine is the second coldest area. The seawater temperatures of other parts of the Black sea are similar with the Marmara Sea. Cluster number 4 covers the north of the Aegean Sea. Cluster number 5 form a great single cluster. The temperature values of the stations in Cluster 6 have also similar characteristics. Cluster number 7 is the hottest region, because it is the closest area to equator. In winter seasons, *C*5 and *C*7 clusters can be marked as one cluster, because they cannot be distinguished very well. In summer seasons, *C*6 cluster becomes a little small. Many factors can effect this distribution of seawater temperature. The temperature varies both attitudinally and depth-wise in response to changes in air-sea interactions. Heat fluxes, evaporation, river in flow, the movement of water and rain all influence the distribution of seawater temperature.

Topex/Poseidon Satellite provides sea surface height residual data as a two-dimensional grid separated by one degree in latitude and longitude. So SSHR values stored in the database are available at 134 stations which are shown in Fig. 8a as black dots. The clusters obtained by the usage of the Sea_Surface_Height table are showed in Fig. 8b. Each cluster has data points that have similar sea surface height residual values. The

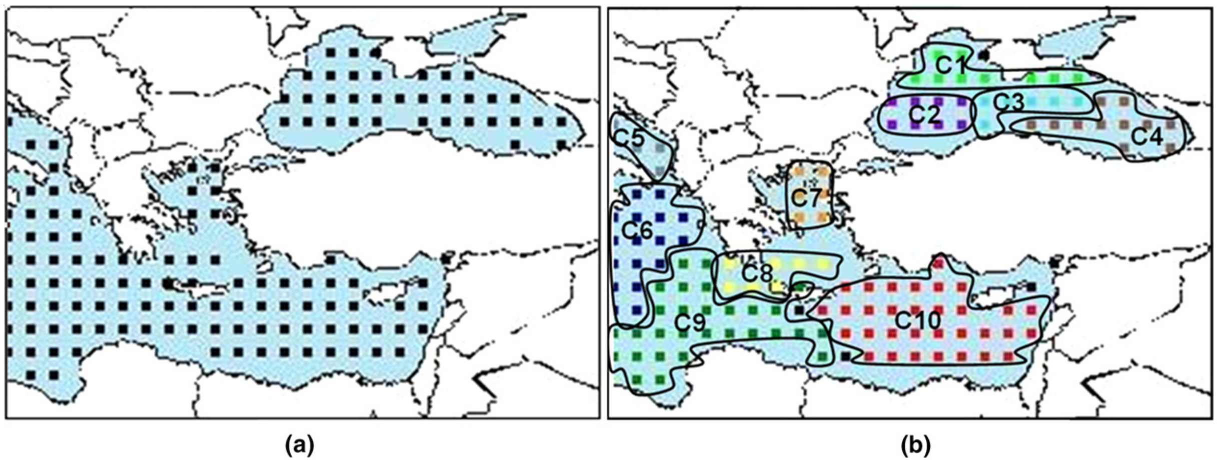

Fig. 8. (a) The locations of 134 stations. (b) The results of cluster analysis on sea surface height residual data.

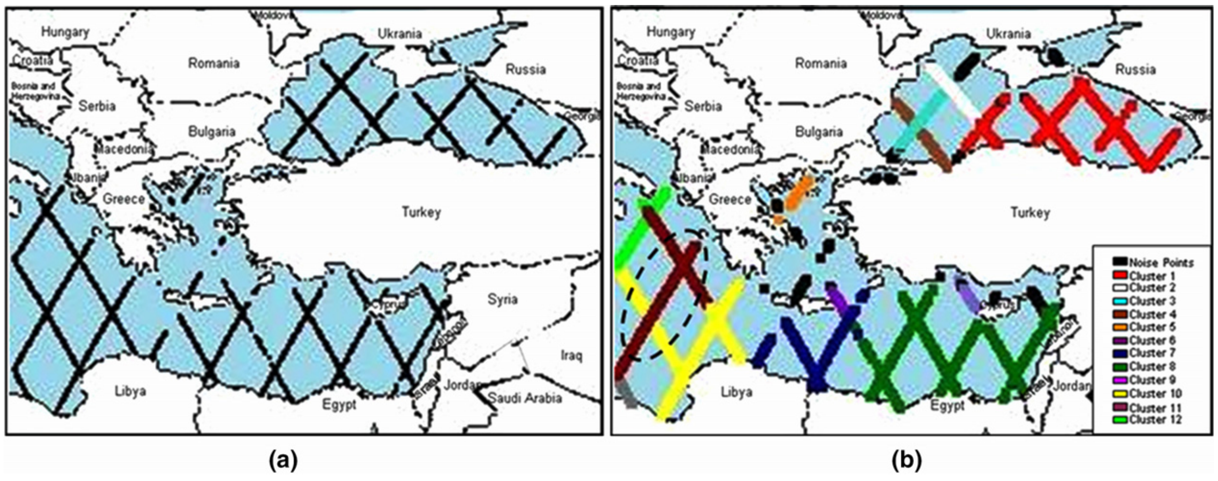

Fig. 9. (a) The locations of 1707 stations. (b) The results of cluster analysis on significant wave height values.

clusters named by *C*1, *C*2, *C*3 and *C*4 are located in Black Sea. The cluster named by *C*7 is located in Aegean Sea. The rest of the clusters are located in Mediterranean Sea. Many factors contribute to changes in sea surface height including sea eddies, temperature of the upper seawater, tides, sea currents, and gravity.

Significant wave height values stored in database were collected at 1707 stations which are shown in Fig. 9a as black dots. Fig. 9b shows an example clustering result obtained by the usage of the dataset measured on January 28, 2001. Each cluster has data points that have similar significant wave height values. For example, while the region at the east of the Crete Island has the wave height values approximately 0.5 m, the region (cluster 11) which is circled in dashed lines has the wave height values approximately 3.6 m. The region which is circled in dashed lines has the maximum wave height values.

6. Conclusions and future work

Clustering is a main method in many areas, including data mining and knowledge discovery, statistics, and machine learning. This study presents a new density-based clustering algorithm ST-DBSCAN which is constructed by modifying DBSCAN algorithm. The first reason of this modification is to be able to discover the clusters on spatial–temporal data. The second modification is necessary to find noise objects when clusters of different densities exist. We introduce a new concept: *density factor*. We assign to each cluster a *density factor*, which is the degree of the density of the cluster. The third modification provides a comparison of the average value of a cluster with new coming value. In order to demonstrate the applicability of our algorithm to real world problems, we present an application using a spatial–temporal data warehouse. Experimental results demonstrate that our modifications appear to be very promising when spatial–temporal data is used to be clustered.

Very large databases need extreme computing power. In future studies, it is intended to run the algorithm in parallel in order to improve the performance. In addition, more useful heuristics may be found to determine the input parameters Eps and MinPts.

Acknowledgements

This study was supported by the Directorate of Scientific Research of the Dokuz Eylul University. The authors are grateful to the Institute of Marine Sciences and Technology at Dokuz Eylul University for help in obtaining the experimental results of this paper.

References

[1] T. Abraham, J.F. Roddick, Survey of spatio-temporal databases, GeoInformatica, Springer 3 (1) (1999) 61–99.
[2] M. Ankerst, M.M. Breunig, H.-P. Kriegel, J. Sander, OPTICS: Ordering points to identify the clustering structure, in: Proceedings of ACM SIGMOD International Conference on Management of Data, Philadelphia, PA, 1999, pp. 49–60.
[3] Z. Aoying, Z. Shuigeng, Approaches for scaling DBSCAN algorithm to large spatial database, Journal of Computer Science and Technology 15 (6) (2000) 509–526.
[4] C. Böhm, S. Berchtold, H.-P. Kriegel, U. Michel, Multidimensional index structures in relational databases, Journal of Intelligent Information Systems (JIIS), Springer 15 (1) (2000) 51–70.
[5] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of Second International Conference on Knowledge Discovery and Data Mining, Portland, OR, 1996, pp. 226–231.
[6] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, Clustering for mining in large spatial databases, KI-Journal (Artificial Intelligence) 12 (1) (1998) 18–24, Special Issue on Data Mining.
[7] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, X. Xu, Incremental clustering for mining in a data warehousing environment, in: Proceedings of International Conference on Very Large Databases (VLDB'98), New York, USA, 1998, pp. 323–333.
[8] D. Fisher, Knowledge acquisition via incremental conceptual clustering, Machine Learning 2 (2) (1987) 139–172.
[9] S. Guha, R. Rastogi, K. Shim, CURE: an efficient clustering algorithms for large databases, in: Proceeding ACM SIGMOD International Conference on Management of Data, Seattle, WA, 1998, pp. 73–84.
[10] R.H. Guting, An introduction to spatial database system, VLDB Journal 3 (4) (1994) 357–399.
[11] A. Guttman, R-trees: a dynamic index structure for spatial searching, in: Proceedings of ACM SIGMOD Int. Conf. on Management of Data, Boston, Massachusetts, 1984, pp. 47–57.
[12] M. Halkidi, Y. Batistakis, M. Vazirgiannis, On clustering validation techniques, Journal of Intelligent Information Systems 17 (2–3) (2001) 107–145.

[13] J. Han, M. Kamber, Data Mining Concepts and Techniques, Morgan Kaufmann Publishers, San Francisco, CA, 2001, pp. 335–391.
[14] J. Han, M. Kamber, A.K.H. Tung, Spatial clustering methods in data mining: a survey, in: H. Miller, J. Han (Eds.), Geographic Data Mining and Knowledge Discovery, Taylor and Francis, London, 2001.
[15] A. Hinneburg, D.A. Keim, An efficient approach to clustering in large multimedia databases with noise, in: Proceedings of 4th International Conference on Knowledge Discovery and Data Mining, New York City, NY, 1998, pp. 58–65.
[16] E. Januzaj, H.-P. Kriegel, M. Pfeifle, Scalable density-based distributed clustering, in: Proceedings of PKDD, Pisa, Italy, Lectures Notes in Computer Science, 3202, Springer, 2004, pp. 231–244.
[17] E. Kolatch, Clustering algorithms for spatial databases: a survey [online]. Available on the web, 2001.
[18] S. Ma, T.J. Wang, S.W. Tang, D.Q. Yang, J. Gao, A new fast clustering algorithm based on reference and density, in: Proceedings of WAIM, Lectures Notes in Computer Science, 2762, Springer, 2003, pp. 214–225.
[19] J. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1967, pp. 281–297.
[20] R.T. Ng, J. Han, Efficient and effective clustering methods for spatial data mining, in: Proceedings of 20th International Conference on Very Large Data Bases, Santiago, Chile, 1994, pp. 144–155.
[21] W.N. Qian, A.Y. Zhou, Analyzing popular clustering algorithms from different view-points, Journal of Software 13 (8) (2002) 1382–1394.
[22] H. Samet, The Design and Analysis of Spatial Data Structures, Addison-Wesley, 1990.
[23] G. Sheikholeslami, S. Chatterjee, A. Zhang, WaveCluster: a multi-resolution clustering approach for very large spatial databases, in: Proceedings of International Conference on Very Large Databases (VLDB'98), New York, USA, 1998, pp. 428–439.
[24] C. Spieth, F. Streichert, N. Speer, A. Zell, Clustering based approach to identify solutions for the inference of regulatory networks, in: Proceedings of the IEEE Congress on Evolutionary Computation, Edinburgh, UK, 2005.
[25] P-N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, Addison-Wesley, 2005.
[26] H. Vinod, Integer programming and the theory of grouping, Journal of the American Statistical Association 64 (326) (1969) 506–519.
[27] C.C. Walton, W.G. Pichel, J.F. Sapper, The development and operational application of nonlinear algorithms for the measurement of sea surface temperatures with the NOAA polar-orbiting environmental satellites, Journal of Geophysical Research 103 (C12) (1998).
[28] W. Wang, J. Yang, R. Muntz, STING: a statistical information grid approach to spatial data mining, in: Proceedings of 23rd International Conference on Very Large Data Bases (VLDB), 1997, pp. 186–195.
[29] J.-R. Wen, J.-Y. Nie, H.-J. Zhang, Query clustering using user logs, ACM Transactions on Information Systems 20 (1) (2002) 59–81.
[30] X. Xu, M. Ester, H.-P. Kriegel, J. Sander, A distribution-based clustering algorithm for mining in large spatial databases, in: Proceedings of IEEE International Conference on Data Engineering, Orlando, FL, 1998, pp. 324–331.
[31] T. Zhang, R. Ramakrishnan, M. Linvy, BIRCH: an efficient data clustering method for very large databases, in: Proceeding ACM SIGMOD International Conference on Management of Data, 1996, pp. 103–114.

Derya Birant received her master degree in computer engineering from Dokuz Eylul University in 2002. Currently, she is a research assistant at the Department of Computer Engineering, Dokuz Eylul University in Turkey. Her research interests include data mining in large databases, data warehousing, parallel computing, web systems modeling and engineering.

Alp Kut is a full professor of computer engineering at Dokuz Eylul University. He is head of the Department of Computer Engineering of Dokuz Eylul University since the fall of 2003. His works include data mining in databases, database management systems and distributed systems. He has many publications on a variety of topics, including, web-based systems and parallel systems.