doi:10.2498/cit.2006.04.04

Spatio-Temporal Outlier Detection in Large Databases

Derya Birant, Alp Kut

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

Outlier detection is one of the major data mining methods. This paper proposes a three-step approach to detect spatio-temporal outliers in large databases. These steps are clustering, checking spatial neighbors, and checking temporal neighbors. In this paper, we introduce a new outlier detection algorithm to find small groups of data objects that are exceptional when compared with the remaining large amount of data. In contrast to the existing outlier detection algorithms, the new algorithm has the ability of discovering outliers according to the non-spatial, spatial and temporal values of the objects. In order to demonstrate the new algorithm, this paper also presents an example of application using a data warehouse.

Keywords: outlier detection, data mining, spatio-temporal data, data warehouse.

1. Introduction

Spatio-temporal databases are growing very rapidly, both in size and in number. This condition results in an increasing need for knowledge discovery in spatio-temporal databases. Most studies in KDD (Knowledge Discovery in Databases) focus on finding the common patterns. However, finding the outliers (rare events or exceptional cases) may be more interesting and useful than finding the common patterns.

Outliers can be defined as observations which appear to be inconsistent with the remainder of the dataset. They deviate too much from other observations. *Outlier detection* is a data mining technique like classification, clustering, and association rules. A *Spatial Outlier* (S-Outlier) is an object whose non-spatial attribute value is significantly different from the values of its spatial neighbors. Recently, a few studies have been conducted on spatial outlier detection for large datasets. [4][16][19] However, most of these studies don't consider temporal aspects. Temporal outlier detection should also be considered in many applications such as geographic phenomena-based applications. A *Temporal Outlier* (T-Outlier) is an object whose non-spatial attribute value is significantly different from those of other objects in its temporal neighborhood. The studies [1][2] focus on the representation of a time location which deviates too much from its temporal neighbors.

This paper combines S-Outlier and T-Outlier definitions to define a *Spatio-Temporal Outlier* (ST-Outlier) to be an object whose non-spatial attribute value is significantly different from those of other objects in its spatial and temporal neighborhoods. For many applications, identification of ST-Outliers can lead to the discovery of unexpected, interesting, and implicit knowledge.

This paper focuses on the question how ST-Outliers can be detected. It presents a new outlier detection algorithm which is based on the DBSCAN clustering algorithm since clustering is a basic method for spatial outlier detection. From the viewpoint of a clustering algorithm, outliers are objects not located in any cluster. Furthermore, if a cluster is significantly different from other clusters, the objects in this cluster might be potential S-Outliers.

The algorithm proposed in this study first identifies S-Outliers and then T-Outliers. However, the identification of first T-Outliers and then S-Outliers yields the same result. So ST-Outliers and TS-Outliers are identical.

The rest of the paper is organized as follows. Section 2 describes related works on the problem of outlier detection. Section 3 explains our algorithm to detect ST-Outliers. Section 4 presents performance evaluation of the algorithm. Section 5 shows the sensitivity analyses of the parameters of the algorithm. Using a real-world dataset, Section 6 presents an application to demonstrate our solution and shows the data mining results. Finally, the conclusion is given in Section 7.

2. Outlier Detection Approaches

The existing approaches to outlier detection can be classified into five categories: *distribution-based*, clustering-based, *depth-based*, *distance-based*, and *density-based* [14][16].

Distribution-based approaches use standard statistical distribution. They deploy some standard distribution model (e.g. Normal, Poisson, etc.) and recognize as outliers those points which deviate from the model. [3] However, for many KDD applications, the underlying distribution is unknown. Often a large number of tests are required in order to decide which distribution model fits the arbitrary dataset best, if any. Fitting the data with standard distributions is costly, and may not produce satisfactory results.

Clustering-based approaches detect outliers as by-products [10]. Some clustering algorithms such as CLARANS [15], DBSCAN [5][6], CURE [7] have the capability of handling exceptions. However, since the main objective of the clustering algorithms is to discover clusters, they are not developed to optimize outlier detection.

Depth-based approaches are based on computational geometry and compute different layers of k-d convex hulls. [11][17] Outliers are more likely to be data objects with smaller depths. However, in practice, this technique becomes inefficient for large datasets ($k \geq 4$). Depth-based approach is also applied for spatial outlier detection. [2]

Distance-based methods use a distance metric to measure the distances among the data points. [12][13] Problems may occur if the parameters of the data are very different from each other in different regions of the data set.

Density-based approach was proposed by M. Breunig, et al. [4]. This method assigns a Local Outlier Factor (LOF) to each sample based on their local neighborhood density. Samples with high LOF value are identified as outliers. The neighborhood is defined by using *MinPts* parameter.

One drawback of the existing methods is that they don't consider temporal aspects. In this paper, we propose a ST-Outlier detection algorithm to overcome this disadvantage. Our algorithm combines the advantages of the clustering-based and density-based approaches.

3. ST-Outlier Detection Algorithm

In our algorithm, a three-step approach is proposed to identify the spatio-temporal outliers. These steps are: clustering, checking spatial neighbors, and checking temporal neighbors.

3.1. Clustering

Clustering is a basic method to detect potential S-Outliers. From the viewpoint of a clustering algorithm, potential outliers are objects not located in any cluster. Furthermore, if a cluster is significantly different from other clusters, the objects in this cluster might be potential S-Outliers.

A clustering algorithm should satisfy three important requirements: (i) discovery of clusters with arbitrary shape (ii) good efficiency on large databases and (iii) some heuristics to determine the input parameters. DBSCAN algorithm satisfies all these requirements. But it doesn't consider temporal aspects and it can't detect some outliers when clusters have different densities. In order to overcome these disadvantages and in order to detect ST-Outliers from a dataset, we improved DBSCAN clustering algorithm in two important directions. The reason of the first modification is to support temporal aspects. In our algorithm, a tree is traversed to find both spatial and temporal neighbors of any object within a given radius. The second modification is necessary to find outliers when clusters have different densities. Our algorithm assigns a *density factor* to each cluster. Density factor is the degree of the density of the cluster. The algorithm also compares the average value of a cluster with the new coming value.

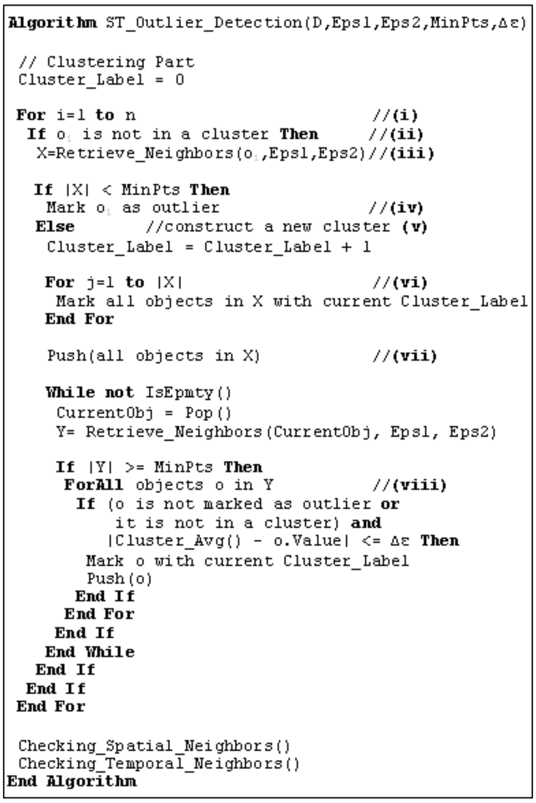
DBSCAN clustering algorithm needs two input parameters to define the notion of density: *Eps* and *MinPts*. The input parameter *Eps* is a radius value and it is based on a distance metric such as Manhattan, Euclidean etc. The second input parameter *MinPts* specifies the minimum number of points that should occur within *Eps* radius. While DBSCAN algorithm needs two inputs, our algorithm requires four input parameters: *Eps1*, *Eps2*, *MinPts*, and $\Delta\varepsilon$. While *Eps1* is the distance parameter for spatial attributes, *Eps2* is the distance parameter for non-spatial attributes. *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 to determine the parameters *Eps* and *MinPts*. The last parameter $\Delta\varepsilon$ is used to prevent the discovering of combined clusters if there are few differences in the values of neighbor locations.

In order to discuss whether a set of points is *similar* enough to be considered a cluster, we need a distance measure Dist(i, j) which tells how far points i and j are. The most common distance measures are Manhattan, Euclidean, and Minkowski distance. In our algorithm, Euclidian formula is used two times to calculate two different distance metrics: *Eps1* (for spatial values) and *Eps2* (for non-spatial values). Euclidean distance is defined as follows:

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

where i=(x_{i1}, x_{i2},..., x_{in}) and j=(x_{j1}, x_{j2},..., x_{jn}) are two n-dimensional data objects.

As shown in Figure 1, the algorithm starts with the first point in database *D*. After processing this point, it selects the next point in *D* (i). If the selected object doesn't 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. If the total number of returned points is smaller than *MinPts* input, the object is assigned as outlier (iv). The points which have been marked as outliers may be changed later. This happens for border points of a cluster.



```
Algorithm ST_Outlier_Detection(D,Eps1,Eps2,MinPts,Δε)

 // Clustering Part
 Cluster_Label = 0

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

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

    For j=1 to |X|                     //(vi)
     Mark all objects in X with current Cluster_Label
    End For

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

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

     If |Y| >= MinPts Then
      ForAll objects o in Y            //(viii)
       If (o is not marked as outlier 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

 Checking_Spatial_Neighbors()
 Checking_Temporal_Neighbors()
End Algorithm
```

Fig. 1. ST-Outlier Detection Algorithm.

If the selected point has enough neighbors within *Eps1* and *Eps2* distances, then a new cluster is constructed (v). Then all neighbors within *Eps1* and *Eps2* radiuses of this object are also marked with new cluster label (vi). Then the algorithm iteratively collects all reachable objects from neighbors by using a stack (vii). If the object is not marked as outlier, 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 $\Delta\varepsilon$, it is placed into the current cluster (viii).

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. *Checking_spatial_neighbors* and *checking_temporal_neighbors* functions are described in sections 3.2 and 3.3.

When the algorithm searches the neighbors of any object by using *Retrieve_Neighbors* function, it takes into consideration both spatial and temporal neighborhoods. The non-spatial value of an object is compared with the non-spatial values of spatial neighbors and also with the values of temporal neighbors (previous day, next day in the same year and the same day in other years).

If two clusters C_1 and C_2 are very close to each other and a point p is the border point of both C_1 and C_2, then the algorithm assigns point p to the cluster discovered first.

3.2. Checking Spatial Neighbors

From the viewpoint of a clustering algorithm, potential outliers are objects not located in any cluster. In the previous step, potential outliers were detected when the data was clustering. In this step, these potential outliers are checked to verify whether these objects are actually S-Outliers or not. During the verification, the background knowledge (the characteristic) of the data is required. If no prior-knowledge about the data is available, some methods such as neural network can be applied to handle it. Furthermore, if a cluster is significantly different from other clusters, the objects in this cluster might be S-Outliers. Thus this step also checks all clusters identified in the previous step to decide whether the cluster is S-Outlier or not. The formula used to verify S-Outliers is defined in definition 1.

Definition 1. Given a database of n data objects $D = \{o_1, o_2, \ldots, o_n\}$. Assume that the object o is detected as potential outlier in clustering. The average value of the spatial neighbors of o within *Eps1* radius is defined as

$$\mathrm{A} \overset{\text{def}}{=} \frac{o_{\text{neigh.1}} + o_{\text{neigh.2}} + \ldots + o_{\text{neigh.m}}}{m} \qquad (2)$$

where m is the number of spatial neighbors of o within *Eps1* radius and the standard deviation for the object o is defined as $\sigma = \sqrt{V}$, where

$$\mathrm{V} \overset{\text{def}}{=} \frac{(o_{\text{neigh1}} - \mathrm{A})^2 + (o_{\text{neigh2}} - \mathrm{A})^2 + \ldots + (o_{\text{neigh m}} - \mathrm{A})^2}{\mathrm{m}}. \qquad (3)$$

The object o is classified as an S-Outlier if it is outside the interval [L,U] (i.e., if either $o < L$ or $o > U$), where $L \overset{\text{def}}{=} A - k_0 \cdot \sigma$, $U \overset{\text{def}}{=} A + k_0 \cdot \sigma$, and $\mathrm{k}_0 > 1$ is some pre-selected value.

3.3. Checking Temporal Neighbors

This step checks the temporal neighbors of the S-Outliers identified in the previous step. 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. During the application of the algorithm, a tree is traversed to find the temporal neighbor objects of any object.

In order to support temporal aspects, S-Outliers are compared to other objects of the same local area, but in different times. In comparison operation, spatio-temporal data is first filtered by retaining only the temporal neighbors and their corresponding values. If the characteristic value of an S-Outlier does not have significant differences from its temporal neighbors, this is not an ST-Outlier. Otherwise, it is confirmed as an ST-Outlier. The formula used to detect ST-Outliers is similar to formula defined in definition 1. In this case temporal neighbors are checked instead of spatial neighbors.

4. Performance Evaluation

The average runtime complexity of the DBSCAN algorithm is O(n*logn), where n is the number of objects in the database. DBSCAN has been proven in its ability of processing very large datasets [5][6]. The algorithm yields significant speed-up factors, even for large numbers of data in the database. The paper [6] shows that the runtime of other clustering algorithms such as CLARANS [15], DBCLASD [20] is between 1.5 and 3 times the runtime of DBSCAN. This factor increases with increasing size of the database. Our modifications do not change the runtime complexity of the algorithm.

As in all databases, fast access to raw data in spatio-temporal databases depends on the structural organization of the stored information and on 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 [18], R-Trees [8] and others, see [9] for an overview. In our study, we made an improvement of the R-Tree indexing method to handle spatio-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.

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. Sensitivity Analyses of the Parameters

Sensitivity analysis is used to determine how a given algorithm output depends upon the input parameters. Most sensitivity analyses involve changing one parameter at a time. A sensitivity analysis was conducted by changing each parameter value by $+/-10\%$. According to the heuristic defined in [5], the input parameters should be assigned as *Eps1=1*, *Eps2=0.25*, and *MinPts=15*. Figure 2 shows the sensitivity of our algorithm to the *Eps1*, *Eps2* and *MinPts* parameters with respect to the number of noise points.

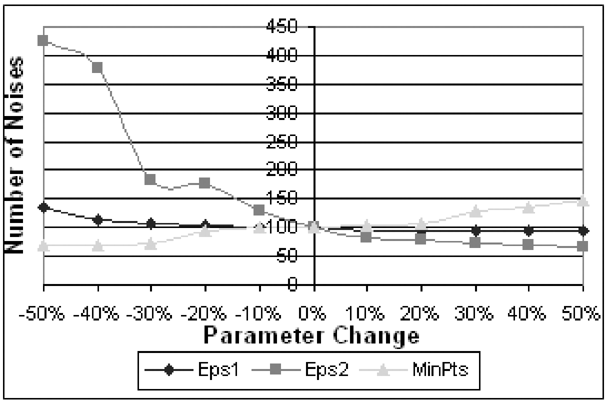


Fig. 2. The sensitivity analyses results.

According to the results of the tests, even where there is a huge variation of *Eps1* parameter, there is a little influence on the results. The number of noise points continuously increases when the value of *Eps1* parameter decreases. But, there is no change in the number of noise points when the value of *Eps1* parameter increases. So *Eps1* parameter has little influence on the algorithm output. The results show that *Eps2* parameter is the most sensitive parameter. Decreasing in the value of the *Eps2* parameter greatly increases the number of noise points. The output of the algorithm is very much depending on the reliability of this parameter. The results also show that the output of the algorithm is little sensitive to changes in the *MinPts* parameter values.

6. Application

This section demonstrates how our algorithm detects ST-Outliers by using a real-world dataset. The purpose of the application is to detect rare events and exceptional cases related with sea waves in years between 1992 and 2002.

6.1. Dataset

We designed a spatio-temporal data warehouse which contained wave height values of four seas: the Black Sea, the Marmara Sea, the Aegean Sea, and the east of the Mediterranean Sea. These seas surround Turkey from the north, west, and south. The geographical coordinates of our work area are 30° to 47.5° north latitude and 17.0° to 42.5° east longitude.

Dataset has been provided from Topex/ Poseidon Satellite [21]. Topex/Poseidon data are released by NASA and CNES. Wave heights are measured in meters. Dataset has approximately six million rows of record. It has the following columns: StationID, RegionID, Year, Month, Day of the record, Latitude of the station, Longitude of the station, WaveHeight value, and ClusterID. Whereas the column, StationID, identifies the geographic location of monitoring station, RegionID identifies the name of the sea, Cluster ID identifies a particular cluster of stations.

6.2. Implementation Details and Results

During the implementation, we first clustered the dataset to find the regions that have similar sea wave height characteristics. The input parameters were designated as $\text{Eps1} = 1$, $\text{Eps2} = 0.25$, and MinPts=15. Second, we checked all noise points and all clusters identified in the previous step to determine S-Outliers.

The region which is circled in dashed lines in Figure 3 had significantly high wave height values on January 24, 1998. The approximate wave height value in this region is 6 meters. So it contains S-Outliers. Third, we checked the temporal neighbors. We compared the wave height values of S-Outliers with other data points of the same location, but in different times. Figure 4 shows the wave height values of the same region in different years. We detected that the region circled in dashed lines in Figure 3 had extreme wave height values in 1998. But, in other years, wave height values of this region were not too high. For this reason, the objects in this region are confirmed as ST-Outliers.

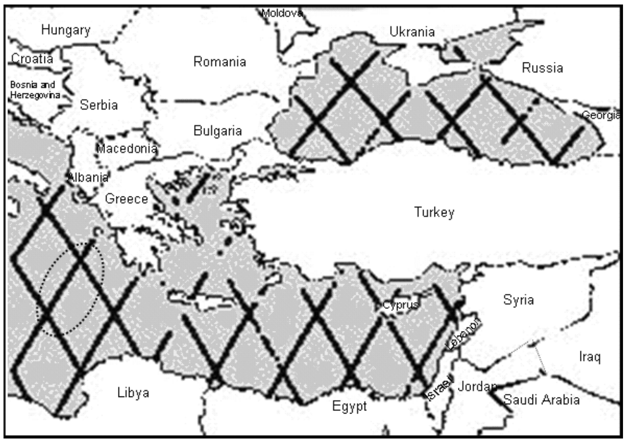


Fig. 3. The region circled in dashed lines contains S-Outliers (January 24, 1998).

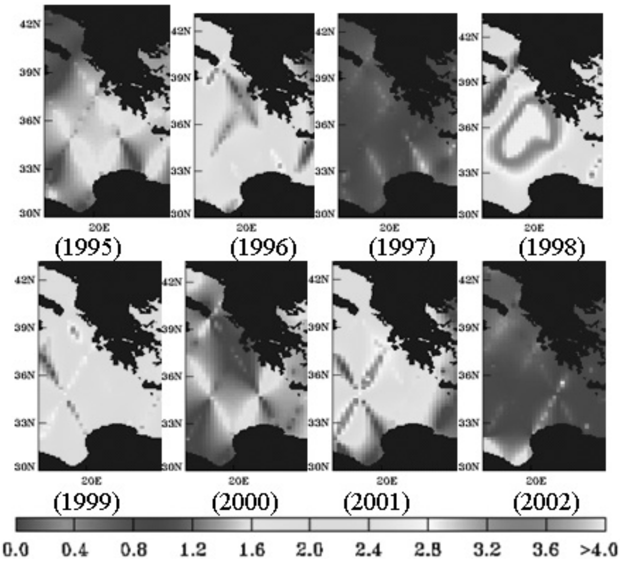


Fig. 4. The wave height values of the same region on the same day, but in different years.

7. Conclusions

This paper proposes a three-step approach to detect spatio-temporal outliers in large databases. These steps are clustering, checking spatial neighbors to identify spatial outliers, and checking temporal neighbors to identify spatio-temporal outliers. The paper introduces a new outlier detection algorithm. According to the performance tests of the algorithm, it has the ability of processing very large datasets. According to the results of the sensitivity analysis of the input parameters, *Eps2* parameter is the most sensitive parameter. The example presented in Section 6 demonstrates that our algorithm appears to be very promising when spatio-temporal outliers need to be detected.

References

[1] T. Abraham, J. F. Roddick, Survey of Spatio-Temporal Databases, *GeoInformatica* (Springer) 1999; 3 (1), pp. 61–99.

[2] N. R. Adam, V. P. Janeja, V. Atluri, Neighbourhood-Based Detection of Anomalies in High Dimension Spatio-Temporal Sensor Datasets, *ACM Symposium on Applied Computing*, Nicosia Cyprus; 2004. pp. 576–583.

[3] V. Barnett, T. Lewis, *Outliers in Statistical Data*, New York: John Wiley; 1994.

[4] M. M. Breunig, H-P. Kriegel, R. Ng, J. Sander, LOF: Identifying Density-Based Local Outliers, *ACM SIGMOD Int. Conf. on Management of Data*, Dallas, TX; 2000, pp. 93–104.

[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 the 2nd Int. Conference on Knowledge Discovery and Data Mining*, Portland, OR; 1996.

[6] M. Ester, H-P. Kriegel, J. Sander, X. Xu, Clustering for Mining in Large Spatial Databases, *KI-Journal (Artificial Intelligence), Special Issue on Data Mining 1998*; 12 (1), pp. 18–24.

[7] S. Guha, R. Rastogi, K. Shim, CURE: An Efficient Clustering Algorithms for Large Databases, In: *Proc. ACM SIGMOD Int. Conf. on Management of Data*, Seattle, WA; 1998. pp. 73–84.

[8] 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.

[9] R. H. GUTING, An Introduction to Spatial Database System, it VLDB Journal, 1994; 3(4), pp. 357–399.

[10] A. JAIN, M. MURTY, P. FLYNN, Data Clustering: A Review, *ACM Computing Surveys*, 1999, 31(3), pp. 264–323.

[11] T. JOHNSON, I. KWOK, R. NG, Fast Computation of 2-Dimensional Depth Contours, In: *Proc. 4th. Int. Conf. on KDD*, New York, NY, 1998, pp. 224–228.

[12] E. M. KNORR, R. T. NG, Algorithms for Mining Distance-Based Outliers in Large Datasets, In: *Proc. 24th Int. Conf. Very Large Data Bases*, New York, NY; 1998, pp. 392–403.

[13] E. M. KNORR, R. T. NG, V. TUCAKOV, Distance-Based Outliers: Algorithms and Applications, Journal: *Very Large Data Bases*, 2000, 8 (3-4), pp. 237–253.

[14] L. KOVÁCS, D. VASS, A. VIDÁCS, Improving Quality of Service Parameter Prediction with Preliminary Outlier Detection and Elimination, In: *Proc. 2nd Int. Workshop on Inter-Domain Performance and Simulation*, Budapest, Hungary, 2004, pp. 194–199.

[15] R. T. NG, J. HAN, Efficient and Effective Clustering Methods for Spatial Data Mining, In: *Proc. 20th Int. Conf. on Very Large Data Bases*, Santiago, Chile, 1994, pp. 144–155.

[16] S. PAPADIMITRIOU, C. FALOUTSOS, Cross-Outlier Detection, In: *Proc. 8th International Symposium on Spatial and Temporal Databases*, Greece, 2003, pp. 199–213.

[17] I. RUTS, P. ROUSSEEUW, Computing Depth Contours of Bivariate Point Clouds, *Journal of Computational Statistics and Data Analysis*, 1996, 23(1996), pp. 153–168.

[18] H. SAMET, *The Design and Analysis of Spatial Data Structures*, MA, Addison-Wesley, 1990.

[19] S. SHEKHAR, C-T. LU, P. ZHANG, A Unified Approach to Detecting Spatial Outliers, *GeoInformatica*, Kluwer Academic Publishers 2003, 7 (2), pp. 139–166.

[20] 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, Florida, 1998. pp. 324–331.

[21] Topex/Poseidon Satellite Description `http://podaac.jpl.nasa.gov/woce/`[15/09/2005]

Received: June, 2006
Accepted: September, 2006

Contact addresses:

Derya Birant
Dokuz Eylul University
Department of Computer Engineering
35100, Izmir
Turkey
`derya@cs.deu.edu.tr`

Alp Kut
Dokuz Eylul University
Department of Computer Engineering
35100, Izmir
Turkey
`alp@cs.deu.edu.tr`

DERYA BIRANT received her PhD in computer engineering from Dokuz Eylul University in 2006. 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 has been head of the Department of Computer Engineering of Dokuz Eylul University since the fall of 2003. His research interests 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.