J. Intell. Syst. **20** (2011), 3–14
DOI 10.1515/JISYS.2011.001

Finding Clusters and Outliers for Data Sets with Constraints

Yong Shi

Abstract. In this paper, we present our research on data mining approaches with the existence of obstacles. Although there are a lot of algorithms designed to detect clusters with obstacles, few algorithms can detect clusters and outliers simultaneously and interactively. We here extend our original research [24] on iterative cluster and outlier detection to study the problem of detecting cluster and outliers iteratively with the presence of obstacles. Clusters and outliers are concepts of the same importance, so it is necessary to treat clusters and outliers in the same way in data analysis. In this algorithm, clusters are detected and adjusted according to the intra-relationship within clusters and the inter-relationship between clusters and outliers, and vice versa. The adjustment and modification of the clusters and outliers are performed iteratively until a certain termination condition is reached. This data processing algorithm can be applied in many fields such as pattern recognition, data clustering and signal processing.

Keywords. Clustering, outlier detection, obstacles.

2010 Mathematics Subject Classification. 62H30.

1 Related Work

More and more large quantities of multi-dimensional data need to be clustered and analyzed. Cluster analysis is used to identify homogeneous and well-separated groups of objects in data sets. It plays an important role in many fields of business and science. The basic steps in the development of a clustering process can be summarized as data cleaning, feature selection, application of a clustering algorithm, validation of results, and interpretation of the results [10] Among these steps, the clustering algorithm and validation of the results are especially critical, and many methods have been proposed in the literature for these two steps. Existing clustering algorithms can be broadly classified into four types [14]: partitioning ([17], [15], [18]), hierarchical ([31], [11], [12]), grid-based ([26], [22] [3]), and density-based ([7] [13], [4]) algorithms. Partitioning algorithms construct a partition of a database of n objects into a set of K clusters, where K is an input parameter. In general, partitioning algorithms start with an initial partition and then use an iterative control strategy to optimize the quality of the clustering re-

sults by moving objects from one group to another. Hierarchical algorithms create a hierarchical decomposition of the given data set of data objects. The hierarchical decomposition is represented by a tree structure, called dendrogram. Grid-based algorithms quantize the space into a finite number of grids and perform all operations on this quantized space. These approaches have the advantage of fast processing time independent of the dataset size and are dependent only on the number of segments in each dimension in the quantized space. Density-based approaches are designed to discover clusters of arbitrary shapes. These approaches hold that, for each point within a cluster, the neighborhood of a given radius must exceed a defined threshold. Density-based approaches can also filter out outliers.

An outlier is a data point that does not follow the main characteristics of the input data. Outlier detection is concerned with discovering the exceptional behaviors of certain objects. It is an important branch in the field of data mining with numerous applications. In some sense it is at least as significant as cluster detection. There are numerous studies on outlier detection. Yu et al. [29] proposed an outlier detection approach termed FindOut as a by-product of WaveCluster ([22]) which removes the clusters from the original data and thus identifies the outliers. Knorr and Ng [16] detected a distance-based outlier which is a data point with a certain percentage of the objects in the data set having a distance of more than $d_{\min}$ away from it. Ramaswamy et al. [19] further extended it based on the distance of a data point from its kth nearest neighbor and identified the top n points with largest kth nearest neighbor distances as outliers. Breunig et al. [6] introduced the concept of local outlier and defined local outlier factor (LOF) of a data point as a degree of how isolated the data point is with respect to the surrounding neighborhood. Aggarwal and Yu [2] considered the problem of outlier detection in subspace to overcome dimensionality curse.

High dimensional data sets continue to pose a challenge to clustering algorithms at a very fundamental level. One of the well-known techniques for improving the data analysis performance is the method of dimension reduction ([3], [1], [21]) in which data is transformed to a lower dimensional space while preserving the major information it carries, so that further processing can be simplified without compromising the quality of the final results. Dimension reduction is often used in clustering, classification, and many other machine learning and data mining applications.

A lot of real data sets have constraint information, and it is difficult to analyze them without considering the existence of the constraints. For example, we might want to find the clusters of office buildings which belong to the same regions based on the locations, but in the real world, there exist many physical obstacles such as rivers, lakes and highways, and their presence may affect the result of clustering substantially.

There are quite a few approaches designed to detect clusters in the presence of obstacles and facilitators. COD_CLARANS [25] is a modified version of the CLARANS [18] partitioning algorithm which performs clustering processes in the presence of obstacles. AUTOCLUST+ [9], is a version of AUTOCLUST [8] enhanced to handle obstacles, which does not require parameters. DBRS+ [28] is derived from DBRS [27], and it handles both obstacles and facilitators. However, none of these algorithms considers detecting outliers simultaneously with clustering process. In many cases, outliers are as important as clusters, such as credit card fraud detection, discovery of criminal activities, discovery of computer intrusion, and etc.

2 Detecting Clusters and Outliers with the Presence of Obstacles

A lot of approaches are designed for clustering and outlier detection methods. We observe that, in many situations, clusters and outliers are concepts whose meanings are inseparable to each other, especially for those data sets with noise. We design a cluster-outlier iterative detection algorithm with the presence of obstacles, tending to detect the clusters and outliers in another perspective for noisy data sets. In this algorithm, clusters are detected and adjusted according to the intra-relationship within clusters and the inter-relationship between clusters and outliers, and vice versa. The adjustment and modification of the clusters and outliers are performed iteratively until a certain termination condition is reached.

Our algorithm is designed to refine and improve the clustering and outlier detection results of clustering algorithms with the presence of obstacles. Let n denote the total number of data points and d be the dimensionality of the data space. Let D_l be the lth dimension, where $l = 1, 2, \ldots, d$. Let the input d-dimensional data set be $X = \{X_1, X_2, \ldots, X_n\}$ which is normalized to be within the hypercube $[0, 1]^d \subset R^d$. Each data point X_i is a d-dimensional vector $X_i = [x_{i1}, x_{i2}, \ldots, x_{id}]$. Data point X_i has the id number i.

There are obstacles existing in the data set as well, which can be represented as multi-dimensional points just like the data points in the data set. If there are k obstacle points:

$$\mathrm{OB} = \{\mathrm{OB}_1, \mathrm{OB}_2, \ldots, \mathrm{OB}_k\}. \tag{1}$$

We normalized those obstacle points within $[0, 1]^d \subset \mathbb{R}^d$, and represent each obstacle point OB_j as a d-dimensional vector:

$$\mathrm{OB}_j = [\mathrm{ob}_{j1}, \mathrm{ob}_{j2}, \ldots, \mathrm{ob}_{jd}]. \tag{2}$$

Each value ob_{jl} where $j = 1, 2, \ldots, k$ and $l = 1, 2, \ldots, d$ represents an obstacle point on dimension D_l where values on the two different sides of ob_{jl} are obstructed so they belong to different segment. Thus, on dimension D_l, where $l = 1, 2, \ldots, d$, the values of all the obstacle points are: ob_{1l}, ob_{2l}, …, ob_{kl} which can be sorted in ascending order: ob'_{1l}, ob'_{2l}, …, ob'_{kl}. The values of ob'_{1l}, ob'_{2l}, …, ob'_{kl} are all within [0, 1] since all the obstacle points are normalized within $[0, 1]^d \subset \mathbb{R}^d$.

Based on the description above, the value range on dimension D_l can be divided into $k + 1$ segments:

$$[0, \mathrm{ob}'_{1l}), [\mathrm{ob}'_{1l}, \mathrm{ob}'_{2l}), \ldots, [\mathrm{ob}'_{kl}, 1] \tag{3}$$

which can be represented as $S_{l0}, S_{l1}, \ldots, S_{lk}$.

According to the input of the initial cluster-outlier division of a data set, we perform our algorithm in an iterative way. In a given iteration step, we assume the current number of clusters is k_c, and the current number of outliers is k_o. The set of clusters is $C = \{C_1, C_2, \ldots, C_{k_\mathrm{c}}\}$, and the set of outliers is $O = \{O_1, O_2, \ldots, O_{k_\mathrm{o}}\}$. We use the term compactness $\mathrm{CPT}(C_i)$ $i = 1, 2, \ldots, k_\mathrm{c}$ to measure the quality of a cluster on the basis of the closeness of data points to the centroid of the cluster. We then define the diversity (distance) between a cluster C and an outlier O, the diversity between two clusters C_1 and C_2, and the diversity between two outliers O_1 and O_2, respectively. Based on these definitions, we measure the quality of a cluster C, the quality of an outlier O, and the quality of the whole data distribution.

The main goal of the obstacle cluster and outlier detection algorithm is to mine the optimal set of clusters and outliers for the input data set. In our approach, clusters and outliers of multi-dimensional data are detected, adjusted and improved iteratively. Clusters and outliers are closely related and they affect each other in a certain way. The basic idea of our algorithm is that clusters are detected and adjusted according to the intra-relationship within clusters and the inter-relationship between clusters and outliers, and vice versa. The adjustment and modification of the clusters and outliers are performed iteratively until a certain termination condition is reached.

The algorithm proceeds in two phases: an initialization phase and an iterative phase. In the initialization phase, we find the centers of clusters and locations of outliers. In the iterative phase, we refine the set of clusters and outliers gradually by optimally exchanging some outliers and some boundary data points of the clusters.

In the initialization phase, we first find the initial set of medoids. In the next step we dispatch data points to appropriate medoids, forming data subsets associated

with medoids. Then we exploit some approaches to determine whether a data subset is a cluster or a group of outliers.

In this phase, given a data point X_i, we need to decide which medoid we dispatch X_i to. Without the existence of obstacle points, we can just simply dispatch X_i to the medoid it is closest to. However, since there are obstacle points, more analysis should be done.

Suppose there are p medoids: M_1, M_2, ..., M_p. On each dimension D_l, $l = 1, 2, \ldots, d$, we first locate the segment S_{lg} $(g = 0, 1, \ldots, k)$ which contains x_{il}. If there is at least one medoid whose value on D_l is also within S_{lg}, we choose the medoid which is closest to x_{il} within S_{lg}. Otherwise, we simply choose the medoid which is closest to x_{il} on D_l. In this way, if there are two medoid M_1 and M_2, and x_{il} is closer to M_1, but not in the same segment with M_1, while it is in the same segment with M_2, we will choose M_2 over M_1 as the medoid for x_{il} on D_l. After we perform this process on all dimensions D_1, D_2, ..., D_d, we select the medoid M_j which is chosen most often for X_i on different dimensions, and dispatch X_i to M_j.

In the iterative phase, we first merge the initial set of clusters into k clusters. In the second step, we sort clusters and outliers based on their qualities and select the worst clusters and outliers. The quality of each cluster is calculated according to the intra-relationship within clusters and the inter-relationship between clusters and outliers, and vice versa. In the third step, for clusters of the worst qualities, we exploit some methods to select the boundary data points for each of them. In the fourth step, we refine the set of clusters and outliers gradually by optimally exchanging the selected boundary data points and the worst outliers. Steps two, three and four are performed iteratively until a certain termination condition is reached.

In step four of this phase, after the exchange of a selected boundary data point *bdp* and a selected outlier o, *bdp* becomes a new outlier and o becomes a new boundary data point. To make sure the exchange is performed in the same segments, we check the segments of *bdp* and o, and exchange them if and only if they are within the same segment.

3 Experimental Results

We conducted comprehensive experiments on both synthetic and real data sets to assess the accuracy and efficiency of the proposed approach. Our experiments were run on Intel(R) Pentium(R) 4 with CPU of 3.99 GHz and Ram of 0.99 GB.

3.1 Experiments on High-Dimensional Data Sets

To test the scalability of our algorithm over dimensionality and data size, we designed a synthetic data generator to produce data sets with normalized distributions. The sizes of the data sets vary from 10,000, 15,000, . . . to 50,000, with the gap of 5,000 between each two adjacent data set sizes, and the dimensions of the data sets vary from 15, 20, . . . to 50, with the gap of 10 between each two adjacent numbers of dimensions.

Figure 1 shows the running time of groups of data sets with dimensions increasing from 15 to 50. Each group has a fixed data size (from 10,000, 15,000, . . . to 50,000).

Figure 2 shows the running time of groups of data sets on one query with sizes increasing from 10,000 to 50,000. Each group has fixed number of dimensions (from 15, 20, . . . to 50).

The two figures indicate that our algorithm is scalable over dimensionality and data size.

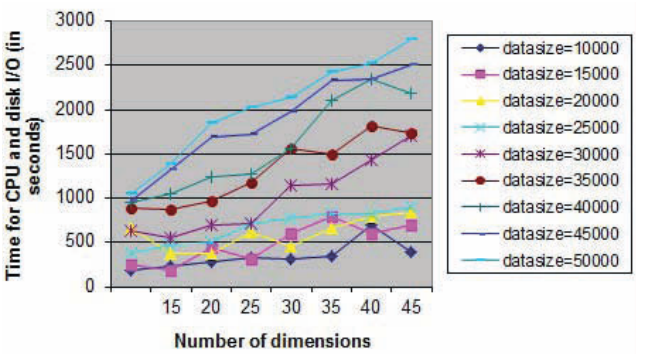

Figure 1. Running time on data sets with increasing dimensions.

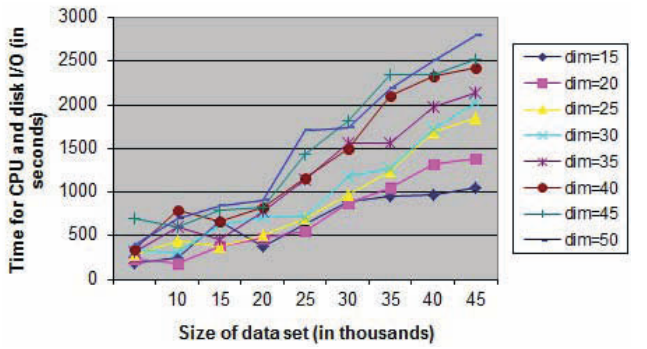

Figure 2. Running time on data sets with increasing data set sizes.

3.2 Experiments on real data sets

We next evaluate the effectiveness of our proposed approach for clustering on real data sets which were obtained from UCI Machine Learning Repository [5]. Here we compare the result of our algorithm with the clustering result of Shrinking algorithm on Wine Recognition data (simplified as Wine data) and Ecoli data.

Wine data set contains the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. It has 178 instances with 13 features. The data set has three clusters with the sizes of 59, 71 and 48.

Ecoli data set contains data regarding Protein Localization Sites. This data set is made up of 336 instances, with each instance having seven features. It contains 8 clusters with the sizes of 143, 77, 52, 35, 20, 5, 2 and 2.

Shrinking algorithm [23] is a data preprocessing technique which optimizes the inner structure of data inspired by the Newton's Universal Law of Gravitation [20] in the real world. Shrinking-based multi-dimensional data analysis approach first moves data points along the direction of the density gradient, thus generating condensed, widely-separated clusters. It then detects clusters by finding the connected components of dense cells. Then it uses a cluster-wise evaluation measurement to compare the clusters from different cases and select the best clustering results.

Table 1 shows the clustering result of Shrinking on Wine data, and Table 2 shows the clustering result of our algorithm on Wine data. We can see that the clustering result is improved after we applied our approach on the Wine data.

Cluster i	1	2	3
$\|C_i^o\|$	59	71	48
$\|C_i^s\|$	53	52	46
$\|C_i^o \cap C_i^s\|$	53	51	43
precision (%)	100	98.08	91.48
recall (%)	89.83	71.83	89.58

Table 1. Clustering results of shrinking algorithm on wine data.

Cluster i	1	2	3
$\|C_i^o\|$	59	71	48
$\|C_i^s\|$	53	65	49
$\|C_i^o \cap C_i^s\|$	53	59	45
precision (%)	100	90.76	91.48
recall (%)	89.83	83.10	93.75

Table 2. Clustering results of our approach on wine data.

	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$
$\|C_i^o\|$	143	77	52	35	20	5	2	2
$\|C_i^s\|$	135	22	68	49	11	N/A	N/A	N/A
$\|C_i^o \cap C_i^s\|$	130	22	43	32	10	N/A	N/A	N/A
precision (%)	96.30	100	63.24	65.31	90.91	N/A	N/A	N/A
recall (%)	90.91	28.57	82.69	91.43	50.00	N/A	N/A	N/A

Table 3. Clustering result of shrinking algorithm for Ecoli data.

	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$
$\|C_i^o\|$	143	77	52	35	20	5	2	2
$\|C_i^s\|$	138	86	71	N/A	N/A	4	N/A	N/A
$\|C_i^o \cap C_i^s\|$	130	55	47	N/A	N/A	4	N/A	N/A
precision (%)	94.20	63.95	66.20	N/A	N/A	100	N/A	N/A
recall (%)	90.90	71.43	90.38	N/A	N/A	80.00	N/A	N/A

Table 4. Clustering result of our approach for Ecoli data.

Experiments are performed also on Ecoli data. We first apply Shrinking-based clustering algorithm on Ecoli data. Table 3 shows the clustering results of the Shrinking algorithm.

Our algorithm is also performed on the Ecoli data. Table 4 shows the clustering results of our algorithm. Compared to the clustering results of the Shrinking algorithm, our approach has the most information of the first 3 largest natural clusters.

3.3 Experiments on More Real Data Sets

We further evaluate the effectiveness of our proposed approach, for finding clusters and outliers in more read data sets. The real data sets were also obtained from UCI Machine Learning Repository [5].

The first data set is the iris data set for various iris plant types. It contains 150 data points, each of which has 4 dimensions. There are 3 classes in the iris data: Iris-setosa, Iris-versicolor, and Iris-virginica.

The second data set is the glass data set for different glass types. It contains 214 data points, each of which has 9 dimensions. There are 7 classes in the glass data, class 1 to class 7.

The third data set is the ionosphere data set which is a radar data set collected by a system in Goose Bay, Labrador. It contains 351 data points, each of which has 34 dimensions. There are two classes in the ionosphere data: g as good, and b as bad.

We conduct experiments on the iris data set. The accuracy rate of clusters and outliers compared to the ground truth of iris data set is 91.2%. We next use the glass data set to test out algorithm. The accuracy rate of clusters and outliers compared to the ground truth of glass data set is 90.3%. We perform the algorithm on the ionosphere data set as well. The accuracy rate of clusters and outliers compared to the ground truth of the ionosphere data set is 93.3%. From the experimental results, we can see our algorithm also performs well on these real data sets in different domains.

4 Conclusion and Discussion

In this paper, we present a novel approach to detect clusters and outlier with the presence of obstacles. We analyze and quantify the relationship among clusters and outliers, and demonstrate the efficiency and effectiveness of our approach.

Bibliography

[1] Aggarwal C., Procopiuc C., Wolf J. et al., Fast Algorithms for Projected Clustering, In: Delis A., Faloutsos C., Ghandeharizadeh S. (eds), Proceedings of the ACM SIGMOD Conference on Management of Data, ACM Press, Philadelphia, PA, USA. pp. 61–72, 1999.

[2] Aggarwal C., Yu P., Outlier Detection for High Dimensional Data. In: Aref W (eds) Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data, ACM Press, Santa Barbara, California, USA, pp. 37–46, 2001.

[3] Agrawal R., Gehrke J., Gunopulos D. et al., Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In: Haas L, Tiwary A (eds) Proceedings of the ACM SIGMOD International Conference on Management of Data, ACM Press, Seattle, WA, USA. pp. 94–105, 1998.

[4] Ankerst M., Breunig M., Kriegel H. et al., OPTICS: Ordering Points To Identify the Clustering Structure, In: Delis A., Faloutsos C., Ghandeharizadeh S. (eds) Proceedings of the ACM SIGMOD Conference on Management of Data, ACM Press, Philadelphia, PA, USA, pp. 49–60, 1999.

[5] Bay S., The UCI KDD Archive [`http://kdd.ics.uci.edu`], University of California, Irvine, Department of Information and Computer Science, Irvine, CA, USA. 1999.

[6] Breunig M., Kriegel H., Ng R. et al., {LOF}: Identifying Density-Based Local Outliers, In: Chen W., Naughton J., Bernstein P. (eds) Proceedings of the ACM SIGMOD CONFERENCE on Management of Data, ACM, Dallas, Texas, USA, pp. 93–104, 2000.

[7] Ester M., Kriegel H., Sander J. et al., A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: Simoudis E, Han J, Fayyad U (eds) Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining. AAAI Press. Portland, OR. pp. 226–231, 1996.

[8] Estivill-Castro V., Lee I., Autoclust: Automatic clustering via boundary extraction for mining massive point-data sets, In: Proceedings of the 5th International Conference on Geocomputation, pp. 23–25, 2000.

[9] Estivill-Castro V., Lee I., Autoclust+: Automatic clustering of point-data sets in the presence of obstacles, In TSDM '00: Proceedings of the First International Workshop on Temporal, Spatial, and Spatio-Temporal Data Mining-Revised Papers, pages 133–146, Springer-Verlag, London, UK, 2001.

[10] Fayyad U., Piatetsky-Shapiro G., Smyth P. et al., Advances in Knowledge Discovery and Data Mining. AAAI Press, Menlo Park, California, USA, 1996.

[11] Guha S., Rastogi R., Shim K., CURE: An Efficient Clustering Algorithm for Large Databases, In: Haas L., Tiwary A. (eds) Proceedings of the ACM SIGMOD International Conference on Management of Data, ACM Press, Seattle, WA, USA, pp. 73-84, 1998.

[12] Guha S., Rastogi R., Shim K., ROCK: A Robust Clustering Algorithm for Categorical Attributes, Proceedings of the IEEE Conference on Data Engineering, IEEE Computer Society Press, Sydney, Australia, pp. 512–521, 1999.

[13] Hinneburg A., Keim D., An Efficient Approach to Clustering in Large Multimedia Databases with Noise. In: Agrawal R, Stolorz P, Piatetsky-Shapiro G (eds) Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, AAAI Press, New York, NY, USA, pp. 58–65, 1998.

[14] Jain A, Murty M, Flyn P., Data Clustering: A Review, ACM Computing Surveys, 31(3) (1999), pp. 264–323.

[15] Kaufman L., Rousseeuw P., Finding Groups in Data: an Introduction to Cluster Analysis, John Wiley \& Sons. Hoboken, NJ, USA, 1990.

[16] Knorr E., Ng R., Algorithms for Mining Distance-Based Outliers in Large Datasets, In: Gupta A., Shmueli O., Widom J. (eds) Proceedings of 24rd International Conference on Very Large Data Bases. Morgan Kaufmann. New York, NY, USA, pp. 392–403, 1998.

[17] MacQueen J., Some methods for classification and analysis of multivariate observations. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. University of California Press, Berkeley, 1:281–297, 1967.

[18] Ng R., Han J., Efficient and Effective Clustering Methods for Spatial Data Mining, In: Bocca J, Jarke M, Zaniolo C (eds) Proceedings of the 20th International Conference on Very Large Data Bases. Morgan Kaufmann. Santiago de Chile, pp. 144–155, 1994.

[19] Ramaswamy S., Rastogi R., Shim K., Efficient Algorithms for Mining Outliers from Large Data Sets, In: Chen W., Naughton J., Bernstein P. (eds) Proceedings of the ACM SIGMOD CONFERENCE on Management of Data, Dallas, Texas, USA, ACM, pp. 427–438, 2000.

[20] Rothman M., The laws of physics. New York, Basic Books, 1963.

[21] Seidl T., Kriegel H., Optimal multi-step k-nearest neighbor search, In Proceedings of the ACM SIGMOD conference on Management of Data, Seattle, WA, pp. 154–164, 1998.

[22] Sheikholeslami G., Chatterjee S., Zhang A., WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases, In: Gupta A., Shmueli O., Widom J. (eds) Proceedings of 24rd International Conference on Very Large Data Bases, Morgan Kaufmann, New York, NY, USA, pp. 428–439, 1998.

[23] Shi Y., Song Y., Zhang A., A Shrinking-Based Approach for Multi-Dimensional Data Analysis, In: Freytag J., Lockemann P., Abiteboul S., et al., Proceedings of 29th International Conference on Very Large Data Bases, ACM, Berlin, Germany, 2003, pp. 440–451.

[24] Shi Y., Zhang A., Towards Exploring Interactive Relationship between Clusters and Outliers in Multi-Dimensional Data Analysis, Proceedings of the 21st International Conference on Data Engineering, IEEE Computer Society, Tokyo, Japan, 2005, pp. 518–519.

[25] Tung A. K. H., Hou J., Han J., Spatial clustering in the presence of obstacles. In: ICDE '01: Proceedings of the 17th International Conference on Data Engineering, p. 359, Washington, DC, USA, IEEE Computer Society, 2001.

[26] Wang W., Yang J., Muntz R., {STING}: A Statistical Information Grid Approach to Spatial Data Mining, In: Jarke M., Carey M., Dittrich K., et al. (eds) Proceedings of 23rd International Conference on Very Large Data Bases, Morgan Kaufmann. Athens, Greece, pp. 186–195, 1997.

[27] Wang X., Hamilton H. J., Dbrs: A density-based spatial clustering method with random sampling. In PAKDD, pp. 563–575, 2003.

[28] Wang X., Rostoker R., Hamilton H. J., Density-Based Spatial Clustering in the Presence of Obstacles and Facilitators, In PaKDD, pp. 446–458, 2004.

[29] Yu D., Sheikholeslami G., Zhang A., FindOut: Finding Outliers in Very Large Datasets. Knowl. Inf. Syst. 4(4) (2002), 387–412.

[30] Zaiane O. R., Lee C. H., Clustering spatial data when facing physical constraints, In In Proc. of the IEEE International Conf. on Data Mining, pp. 737–740, 2002.

[31] Zhang T., Ramakrishnan R., Livny M., BIRCH: An Efficient Data Clustering Method for Very Large Databases. In: Jagadish H., Mumick I. (eds.) Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, ACM, Montreal, Quebec, Canada, pp. 103–114, 1996.

Received September 10, 2010.

Author information

Yong Shi, Department of Computer Science and Information Systems,
Kennesaw State University, Building 11, Room 3060, Kennesaw, GA 30144, USA.
E-mail: yshi5@kennesaw.edu