Global Journal of Computer Science and Technology
Software & Data Engineering
Volume 12 Issue 13 Version 1.0 Year 2012
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Performanace of Improved Minimum Spanning Tree Based on Clustering Technique

By P.Sampurnima, J Srinivas & Harikrishna
NOVA college of engineering for women Vijayawada

Abstract - Clustering technique is one of the most important and basic tool for data mining. Cluster algorithms have the ability to detect clusters with irregular boundaries, minimum spanning tree-based clustering algorithms have been widely used in practice. In such clustering algorithms, the search for nearest objects in the construction of minimum spanning trees is the main source of computation and the standard solutions take $O(N^2)$ time. In this paper, we present a fast minimum spanning tree-inspired clustering algorithm, which, by using an efficient implementation of the cut and the cycle property of the minimum spanning trees, can have much better performance than $O(N^2)$.

Keywords : Clustering, graph algorithms, Minimum spanning tree, Divisive hierarchical clustering algorithm.

GJCST-C Classification: E.1

PERFORMANACE OF IMPROVED MINIMUM SPANNING TREE BASED ON CLUSTERING TECHNIQUE

Performanace of Improved Minimum Spanning Tree Based on Clustering Technique

P.Sampurnima [α], J Srinivas [σ] & Harikrishna [ρ]

Abstract - Clustering technique is one of the most important and basic tool for data mining. Cluster algorithms have the ability to detect clusters with irregular boundaries, minimum spanning tree-based clustering algorithms have been widely used in practice. In such clustering algorithms, the search for nearest objects in the construction of minimum spanning trees is the main source of computation and the standard solutions take O(N^2) time. In this paper, we present a fast minimum spanning tree-inspired clustering algorithm, which, by using an efficient implementation of the cut and the cycle property of the minimum spanning trees, can have much better performance than O(N^2).

Keywords : Clustering, graph algorithms, Minimum spanning tree, Divisive hierarchical clustering algorithm.

I. Introduction

Given a set of data points and a distance measure, clustering is the process of partitioning the data set into subsets, called clusters, so that the data in each subset share some properties in common. Usually, the common properties are quantitatively evaluated by some measures of the optimality such as minimum intracluster distance or maximum intercluster distance, etc.

Clustering, as an important tool to explore the hidden structures of modern large databases, has been extensively studied and many algorithms have been proposed in the literature. Because of the huge variety of the problems and data distributions, different techniques, such as hierarchical, partitional, and density- and model-based approaches, have been developed and no techniques are completely satisfactory for all the cases.

For example, some classical algorithms rely on either the idea of grouping the data points around some "centers" or the idea of separating the data points using some regular geometric curves such as hyper planes. As a result, they generally do not work well when the boundaries of the clusters are irregular. Sufficient empirical evidences have shown that a minimum spanning tree representation is quite invariant to the detailed geometric changes in clusters' boundaries.

Author α : M.Tech Student Department of Computer Science and Engineering NOVA college of engineering for women Vijayawada. E-mail : psampurnima@gmail.com
Author σ : Associate professor Department of Computer Science and Engineering NOVA college of engineering for women Vijayawada.
Author ρ : HOD Department of Computer Science and Engineering NOVA college of engineering for women Vijayawada.

Therefore, the shape of a cluster has little impact on the performance of minimum spanning tree (MST)-based clustering algorithms, which allows us to overcome many of the problems faced by the classical clustering algorithms.

II. An MST-Inspired Clustering Algorithm

Although MST-based clustering algorithms have been widely studied, in this section, we describe a new divide and- conquer scheme to facilitate efficient MST-based clustering in modern large databases. Basically, it follows the idea of the "Reverse Delete" algorithm. Before proceeding, we give a formal proof of its correctness.

Theorem 1. *Given a connected, edge-weighted graph, the "Reverse Delete" algorithm produces an MST.*

Proof. First, we show that the algorithm produces a spanning tree. This is because the graph is given connected at the beginning and, when deleting edges in the non increasing order, only the most expensive edge in any cycle is deleted, which does eliminate the cycles but not disconnect the graph, resulting in a connected graph containing no cycle at the end. To show that the obtained spanning tree is an MST, consider any edge removed by the algorithm. It can be observed that it must lie on some cycle (otherwise removing it would disconnect the graph) and it must be the most expensive one on it (otherwise retaining it would violate the cycle property). Hence, the "Reverse Delete" algorithm produces an MST.

For our MST-inspired clustering problem, it is straightforward that n=N and m=N (N-1)/2, and the standard solution has O (N^2logN) time complexity. However, m=O (N^2) is not always necessary. The design of a more efficient scheme is motivated by the following observations. First, the MST-based clustering algorithms can be more efficient if the longest edges of an MST can be identified quickly before most of the shorter ones are found. This is because, for some MST-based clustering problems, if we can find the longest edges in the MST very quickly, there is no need to compute the exact distance values associated with the shorter ones.

Second, for other MST-based clustering algorithms, if the longest edges can be found quickly, the Prim's algorithm can be more efficiently applied to each individual size-reduced cluster. This divide-and-conquer approach will allow us to save the number of distance computations tremendously.

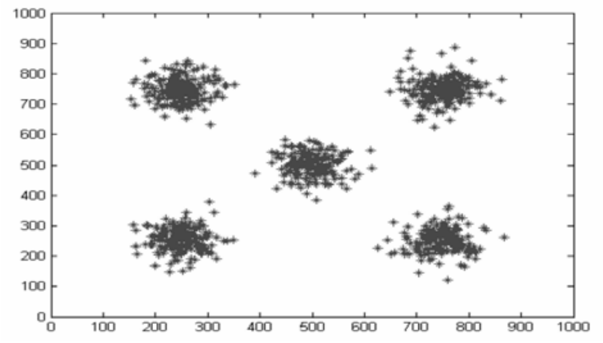


Figure 1 : A two-dimensional five-cluster data set

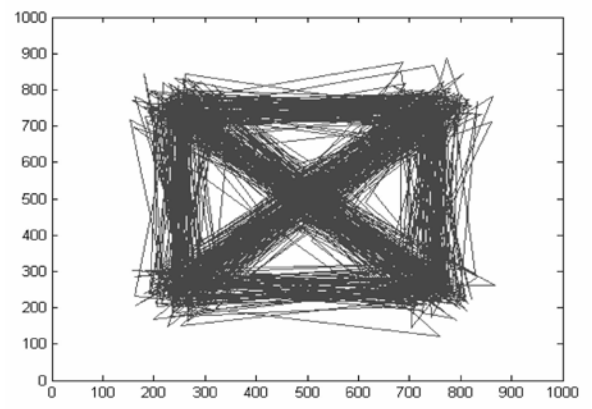


Figure 2 : Its spanning tree after the sequential initialization

a) A Simple Idea

Given a set of S-dimensional data, i.e., each data item is a point in the s-dimensional space, there exists a distance between every pair of the data items. To compute all the pairwise distances, the time complexity is O (sN^2), where N is the number of data items in the set. Suppose at the beginning, each data item is initialized to have a distance with another data item in the set. For example, since the data items are always stored sequentially, each data item can be assigned the distance between itself and its immediate predecessor—called a forward initialized tree—or successor—called a backward initialized tree. These initial distances, whatever they are, provide an upper bound for the distance of each data item to its neighbor in the MST.

In the implementation, the data structure consists of two arrays:

1. Distance array
2. Index array.

Distance Array:

The distance array is used to record the distance of each data point to some other data point in the sequentially stored data set.

Index Array:

The index array records the index of the data item at the other end of the distance in the distance array.

According to the working principle of the MST-based clustering algorithms, a database can be split into partitions by identifying and removing the longest inconsistent edges in the tree. Based on this finding, after the sequential initialization, we can do a search in the distance array (i.e., the current spanning tree) for the edge that has the largest distance value, which we call the potential longest edge candidate. Then the next step is to check whether or not there exists another edge with a smaller weight crossing the two partitions connected now by this potential longest edge candidate. If the result shows that this potential longest edge candidate is the edge with the smallest weight crossing the two partitions, we find the longest edge in the current spanning tree (ST) that agrees with the longest edge in the corresponding MST. Otherwise, we record the update and start another round of the potential longest edge candidate identification in the current ST.

It can be seen that the quality of our fast algorithm depends on the quality of the initialization to quickly expose the longest edges. Though the sequential initialization gives us a spanning tree, when the data are randomly stored, such a tree could be far from being optimal. This situation can be illustrated by a two-dimensional five cluster data set shown in Figure. 1. Shown in Figure.2 is its spanning tree after the sequential initialization (SI). In order to quickly identify the longest edges, we propose to follow the sequential initialization by multiple runs of a recursive procedure known as the divisive hierarchical clustering algorithm (DHCA).

b) Divisive Hierarchical Clustering Algorithm

Essentially, given a data set, the DHCA starts with k randomly selected centers and then assigns each point to its closest center, creating k partitions. At each stage in the iteration, for each of these k partitions, DHCA recursively selects k random centers and continues the clustering process within each partition to form at most k^n partitions for the nth stage. In our implementation, the procedure continues until the number of elements in a partition is below k+2, at which time, the distance of each data item to other data items in that partition can be updated with a smaller value by a brute-force nearest neighbor search. Such a strategy ensures that points that are close to each other in space are likely to be collocated in the same partition. However, because any data point in a partition is closer to its cluster center (not its nearest neighbor) than to the center of any other partition (in case, the data point is equidistant to two or more centers, the partition to which the data point belongs is a random one), the data points in the cluster's boundaries can be misclassified into a wrong partition. Fortunately, such possibilities can be greatly reduced by multiple runs of DHCA. To summarize, we believe that the advantage of DHCA is that, after multiple runs, each point will be very close to its true nearest neighbor in the data set.

To demonstrate this fact, one can think of this problem as a set of independent Bernoulli trials where one keeps running DHCA and classifying each data point to its closest randomly selected cluster center at each stage of the process, until it succeeds (i.e., it hits its nearest neighbor, or at least, its approximate nearest neighbor). Let p be the probability that a random data point hits its nearest neighbor. Let Y be the random variable representing the number of trials needed for a random data point to hit its nearest neighbor. The probability of obtaining a success on trial y is given by

$$P(Y=y)=q^{y-1}p,$$

Where q=1-p denotes the probability that a failure occurs. The relationship between p and P(Y=y) is plotted in from it, we can see that for a randomized process (i.e., p=0.5), at most 50 DHCAs are enough for most of the data points to meet their nearest neighbor.

For our purpose, after the sequential initialization, a spanning tree is constructed and each data item in the tree has already had a distance. During the divisive hierarchical clustering process, each data item will have multiple distance computations.

c) MST-Inspired Clustering Algorithm

Based on the methodology presented in the previous two sections, given a loose estimate of minimum and maximum numbers of data items in each cluster, an iterative approach for our MST-inspired clustering algorithm can be summarized in the following:

1. Start with a spanning tree built by the SI.
2. Calculate the mean and the standard deviation of the edge weights in the current distance array and use their sum as the threshold. Partially refine the spanning tree by running our DHCA multiple times until the percentage threshold difference between two consecutively updated distance arrays is below 10^{-6}.
3. Identify and verify the longest edge candidates by running MDHCA until two consecutive longest edge distances converge to the same value at the same places.
4. Remove this longest edge.
5. If the number of clusters in the data set is preset or if the difference between two consecutively removed longest edges has a percentage decrement larger than 50 percent of the previous one, we stop. Otherwise go to Step 3.

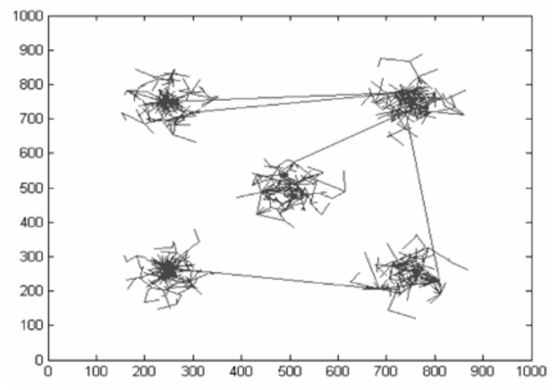


Figure 3 : Updated spanning tree using DHCAs

We stop Step 2 when the percentage threshold difference between two consecutive pruning thresholds, i.e., its percentage decrement, is below a threshold, say 10^{-6} in our implementation, because further DHCA-based distance upper bound updates will not bring us more gains which are worth the overhead of the DHCA. The spanning tree after the DHCA updates for the one shown in Figure. 2 is manifested in Figure. 3.

The terminating condition presented in the above MST inspired clustering algorithm is under the assumptions that the clusters are well separated and there are no outstanding outliers. However, in many real-world problems, the clusters are not always well separated and noise in the form of outliers often exists. For these cases, some of the longest edges do not correspond to any cluster separations or breaks but are associated with the outliers for such cases, we propose terminating

Conditions of that are adaption results from LM algorithm and the MSDR algorithm.

The advantage of the LM algorithm is the avoidance of unnecessary large number of small clusters. The advantage of the MSDR algorithm is that it can find the optimal cluster separations, particularly for cases where there exist some unknown hidden structures in the data set.

The adapted LM algorithm is the following:

1. Get a loose estimate of the maximum and Minimum number of data points for each cluster.
2. Always cut the largest subcluster and cut an edge only when the sizes of both clusters resulted by cutting that edge are larger than the minimum number of data points.
3. Terminates when the size of the largest cluster becomes smaller than the estimated maximum number of data points.

The adapted LM algorithm is the following:

1. Calculate the mean and the standard deviation of the edge weights in the distance array and use their sum as the threshold. 'Remove the longest edge that is larger than the threshold and that links either

a single point or a very small number of data points to the MST.

2. Continue Steps 1 and 2 until the edge is reached, by removing which, two large groups will form from the single largest group before that edge is cut.
3. Apply the MSDR algorithm on the denoised MST
4. Assign the removed data points the same cluster Label as their nearest neighbor's.

d) *Time Complexity Analysis*

From the description in the previous sections, it can be seen that our algorithm mainly consists of two phases. The first phase includes the sequential initialization and the DHCA spanning tree updating, and the second phase uses the MDHCA to locate the longest edges and partitions the obtained approximate minimum spanning tree to form sensible clusters. We expect the original DHCAs (i.e., no thresholding involved) to scale as O (fN logN), where f denotes the number of DHCA constructed before the terminating condition is satisfied. Since in our implementation, at each step of the spanning tree updating using the DHCA, before we assign a data item to a cluster center, if its current distance upper bound is smaller than the threshold (i.e., the sum of the mean and one standard deviation of the tree edge weights), we ignore it, the time complexity is actually (d(xN)log(xN)), where x is between 0 and 1.

Therefore, as long as x is small enough, the time complexity could be near linear on average. Though its worst time complexity could be $O(N^2)$, the average time complexity of the second phase is O(eN logN), where e denotes the number of MDHCA constructed before the terminating condition is satisfied. Since, on average, the number of longest edges is much smaller than the data set size N, as long as the spanning tree constructed in the first phase is very close to the true minimum spanning tree, we expect our MST-inspired algorithm to scale as O(log N).

e) *Pseudocode for Our Clustering Algorithm*

The implementation of the DHCA in our approach is through the design of a C++ data structure called Node. The Node data structure has several member variables that remember the indexes of the subset of the data items that are clustered into it from its parent level, the indexes of its randomly chosen k cluster centers from its own set for its descendants, and a main member function that generates k new nodes by clustering its own set into k sub clusters. The outputs of the Node data structure are at most k new Nodes as the descendents of the current one.

The divisive hierarchical clustering process starts with creating a Node instance, called the top Node. This top Node has every data item in the data set as its samples. From these samples, this top Node randomly chooses k data points as its clustering centers and assigns each sample to its nearest one, generating k data subsets in the form of k Nodes. Only when the number of samples in a Node is larger than a predefined cluster size will that Node be pushed to the back of the topNode, forming an array of Nodes. This process continues recursively. With the new Nodes being generated on the fly and pushed to the back of the Node array, they will be processed in order until no new Nodes are generated and the end of the existing Node array is reached.

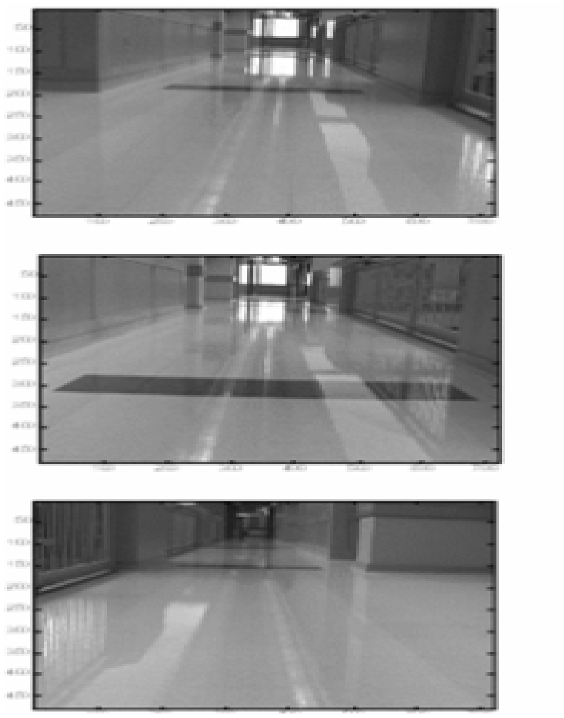
Totally, we need two variants of the DHCA procedure, DHCA for our spanning tree updating, and MDHCA for the cycle property implementation. The DHCA_ST procedure is given in Table 1. The DHCA_CYC procedure is the same as DHCA_ST except for the ways to choose cluster centers and will not be repeated here.

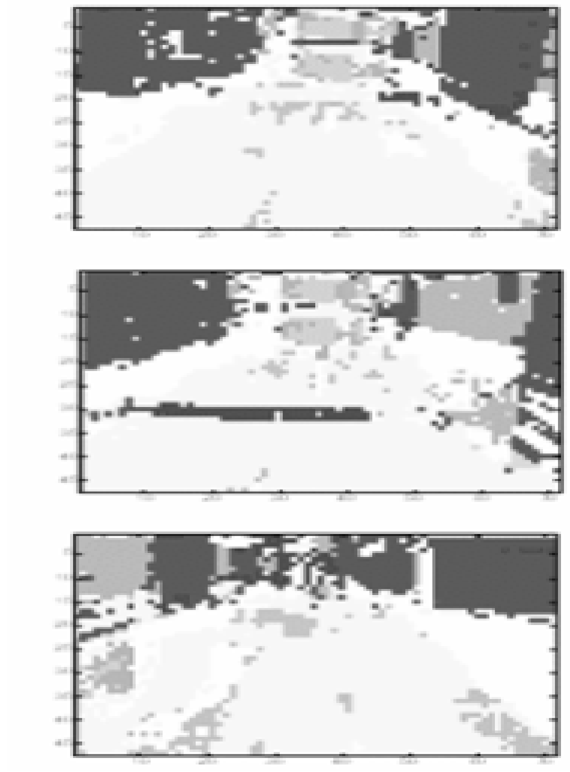
Procedure Name	DHCA_ST
Input:	
Dist_st, edge_st array	The ST distance array and index
Dist_knn,edge_knn Nearest	The auxiliary arrays to remember k-Neighbours(kNN) for each data item
kNN	The no.of NNs of a data item
nodeArray	An array of the Node structures
currentNode	The current Node in the Node array
k	The number of clusters at each step
data	The input data set
maxclustersize	The maximum size of each clusters
threshold	The value used to filter

```
Output:
 Updated dist_st,edge_st,dist_knn,edge_knn and new generated<=k
Nodes which are pushed to the back of nodeArray
Begin
    Randomly select k centers from sampleNumbers of currentNode;
    Generate k newNodes;
    For each sample i in sampleNumbers of currentNode that is not a center
    {
        find its nearest center j out of k;
        if((dist_st[i]<distance(i , j)&&
                (sampleNumber[i]>sampleNumber[j]))
          {
             Update dist_st, edge_st;
          }
          if(dist_knn[i].max>distance(i,j))
          {
             Update dist_knn, edge_knn;
          }
          if(dist_st[i]>threshold)
          {
      assign sampleNumbers[i] to groups of center j;
    }
  }
   for each newNode j=1 to k
   {
          if(newNode[j].sampleNumbers.size()>maxclustersize)
```

```
            {
                Push newNode[j] to the end of nodeAyyay;
             }
        }
   End
```

dist_knn[i].max is the kNNth nearest neighbor of data item i.


(a) Original image


(b) On our MST

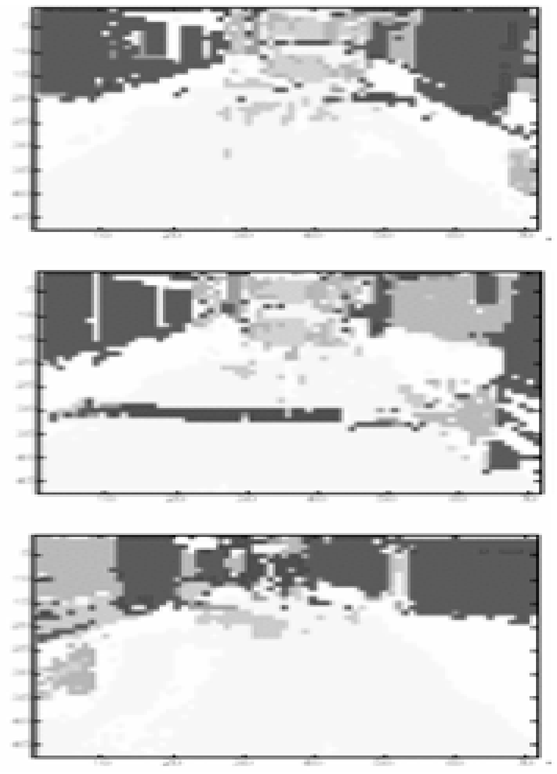

(c) On Prim's MST

Figure (a-c) : Results of the adapted MSDR algorithm

We conducted extensive experiments to evaluate our algorithm against the k-means algorithm and two other state-of-the-art MST-based clustering algorithms on three standard synthetic data sets and two real data sets. The experimental results show that our proposed MST inspired clustering algorithm is very effective and stable when applied to various clustering problems. Since there often exist some structures in the data sets, our algorithm does not necessarily require but can automatically detrmine the desired number of clusters by itself.

In the future, we will further study the rich properties of the existing MST algorithms and adapt our proposed MST inspired clustering algorithm to more general and larger data sets, particularly when the whole data set cannot fit into the main memory.

III. Conclusion

As a graph partition technique, the MST-based clustering algorithms are of growing importance in detecting the irregular boundaries. A central problem in such clustering algorithms is the classic quadratic time complexity on the construction of an MST. In this paper, we present a more efficient method that can quickly identify the longest edges in an MST so as to save some computations. Our contribution is the design of a new MST-inspired clustering algorithm for large data sets (however, without any specific requirements on the distance measure used) by utilizing a DHCA in an efficient implementation of the cut and the cycle property.

References Références Referencias

1. I. Katriel, P. Sanders, and J.L. Traff, "A Practical Minimum Spanning Tree Algorithm Using the Cycle Property," Proc. 11th European Symp. Algorithms (ESA '03), vol. 2832, pp. 679-690, 2003.
2. C.T. Zahn, "Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters," IEEE Trans. Computers, vol. 20, no. 1, pp. 68-86, Jan. 1971.
3. A. Vathy-Fogarassy, A. Kiss, and J. Abonyi, "Hybrid Minimal Spanning Tree and Mixture of Gaussians Based Clustering Algorithm," Foundations of Information and Knowledge Systems, pp. 313-330, Springer, 2006.
4. O. Grygorash, Y. Zhou, and Z. Jorgensen, "Minimum Spanning Tree-Based Clustering Algorithms," Proc. IEEE Int'l Conf. Tools with Artificial Intelligence, pp. 73-81, 2006.
5. R.C. Gonzalez and P. Wintz, Digital Image Processing, second ed. Addison-Wesley, 1987.
6. Y. Xu, V. Olman, and D. Xu, "Clustering Gene Expression Data Using a Graph-Theoretic Approach: An Application of Minimum Spanning Trees," Bioinformatics, vol. 18, no. 4, pp. 536-545, 2002.
7. J. Kleinberg and E. Tardos, *Algorithm Design,* pp.142-149. Pearson-Addison Wesley, 2005.
8. A. Ghoting, S. Parthasarathy, and M.E. Otey, "Fast Mining of Distance-Based Outliers in High Dimensional Data Sets," *Proc. SIAM Int'l Conf. Data Mining (SDM),* vol. 16, no. 3, pp.349-364, 2006.
9. I. Katriel, P. Sanders, and J.L. Traff, "A Practical Minimum Spanning Tree Algorithm Using the Cycle Property," *Proc. 11th European Symp. Algorithms (ESA '03),* vol. 2832, pp.679-690, 2003.
10. J. Lin, D. Ye, C. Chen, and M. Gao, "Minimum Spanning Tree-Based Spatial Outlier Mining and Its Applications," *Lecture Notes in Computer Science,* vol. 5009/2008, pp.508-515, Springer-Verlag, 2008.