*J. Service Science & Management,* **2009, 2: 362-367**
doi:10.4236/jssm.2009.24043 Published Online December 2009 (www.SciRP.org/journal/jssm)

# A Projection Clustering Technique Based on Projection

**Xiyu LIU[1], Xinjiang XIE[2], Wenping WANG[1]**

[1]School of Management and Economics, Shandong Normal University, Jinan, China; [2]Department of Computer Science and Technology, Shandong Agricultural Administrators College, Jinan, China.
Email: xyliu@sdnu.edu.cn, wenping216@126.com, xjxie@163.com

Received July 24, 2009; revised September 10, 2009; accepted October 20, 2009.

## ABSTRACT

*Projection clustering is an important cluster problem. Although there are extensive studies with proposed algorithms and applications, one of the basic computing architectures is that they are all at the level of data objects. The purpose of this paper is to propose a new clustering technique based on grid architecture. Our new technique integrates minimum spanning tree and grid clustering together. By this integration of projection clustering with grid technique, the complexity of computing is lowered to* $O(NlogN)$.

***Keywords*: *Cluster Analysis, Projection Clustering, Grid Clustering***

## 1. Introduction

Cluster analysis is as an important area in data mining which can explore the hidden structures of business databases [1]. Traditionally, cluster analysis can be categorized as three classes. Partitioning method works by constructing various partitions and then evaluating them by some criterion. Hierarchy method creates a hierarchical decomposition of the set of data (or objects) using some criterion. Density-based method is based on connectivity and density functions. Grid-based method is based on a multiple-level granularity structure. Model-based method is to construct a model and to find the best fit model.

Along these lines, many techniques and algorithms have been proposed in the literature. For example, Ester *et al.* [2] present the density-based clustering algorithm which uses an Eps-neighborhood for a point containing at least a minimum number of points. Raphael Bar-Or and Christiaan van Woudenberg [3,4] present a gravity-based clustering method intended to find clusters of data in n-space. The most classical clustering technique is due to Raymond T. Ng and Jiawei Han [5] who developed a CLARANS which aims to use randomized search to facilitate the clustering of a large number of objects. More recent work include agglomerative fuzzy K-Means clustering algorithm by introducing a penalty term to the objective function to make the clustering process insensitive to the initial cluster centers [6].

Among all these clustering techniques, one of the basic measurements is the Euclidean distance. It requires similar objects to have close values in all dimensions. When similarity between objects in high dimensional space is absent, this kind of technique is often invalid. To solve this problem, dimension reduction and manifold learning is applied [7–9]. Another method for this skewed data is the projection clustering [10]. The main idea of projected clustering is that different clusters may distribute along part of the dimensions. A projected cluster is a subset of data points, together with a subspace of dimensions, so that the points are closely clustered in the subspace.

Different with the above clustering approaches, graph clustering works by transforming the initial working data into a kind of graph. Then graph clustering techniques can be applied to obtain the final clustering. One of these techniques is the Minimum Spanning Tree (MST) based clustering. Although the first MST-based clustering algorithms have been studied for many years, due to its computational efficiency for large databases, it attracts new researches frequently. In a more recent work [11], the authors present a more efficient method based on the divide and conquer approach that can quickly identify the longest edges in an MST so as to save some computations. The experimental results show that their MST inspired clustering algorithm is very effective and stable when applied to various clustering problems. The authors also expect that their algorithms have a $O(N \log N)$ computing time.

In this paper, we propose a new projection clustering technique by Minimum Spanning Tree based on the grid

 

clustering approach. Basically, our MST-inspired clustering technique works on cells rather than data points directly. This will significantly reduce the size of graph and MST. Due to this reason, our technique has no specific requirements on the dimensionality of the data sets. This is different from some typical projection clustering algorithms [10].

The rest of this paper is organized as follows: Section 2 presents the basic idea of projection clustering. In Section 3, we summarize some basics on MST-based clustering techniques. In Section 4, we propose a flexible grid clustering approach. Clustering is also discussed as an optimization process in this section. Section 5 contains a brief clustering behavior and time complexity analysis.

## 2. Projection Cell Clustering

Suppose the raw data set is $X = \{x_1, \cdots, x_N\}$. Each point has *n* components by $x = (x^1, \cdots, x^n)$. It is contained in a rectangle $D_0$ in $R^n$. Generally, we will not cluster the original data set *X*. Instead, we will consider a derived data set *X* composed of data cells. However, after we transforme the data set into cells, each cell can be considered as a new data point represented by its center. The number of data points in the cell is called the mass of the cell.

More precisely, a cell (grid) is a polyhedron as a complex *x* = (*D, norm, c, b, p*), where *D* is the polyhedron, *norm* is a unit vector indicating normal vector of one of its edges, *c* is the center, *b* is a boolean value indicating whether the grid is dense or sparse, and *p* is the number of data points covered by the grid.

In order to simplify symbols, we will use $x \in \mathfrak{X}$ as cell data object and $[x] \subset X$ as the data points defined by *x* in the original data space *X*. For two objects *x, y*, the distance is defined as the minimum distance of the two cells

$$\rho(x, y) = \min_{p \in [x], q \in [y]} d(p, q) \tag{1}$$

The diameter of a data object is measurement of its size

$$\sigma(x) = \frac{1}{2} \max_{p,q \in [x]} \rho(p, q) \tag{2}$$

Let N(x) be the set of k-nearest neighbors (KNN) of x including itself, then the number of object in N(x) is k+1. The sparseness or thickness of the data object can be measured by the relative location of its k-nearest neighbors

$$\mu(x) = \frac{1}{k} \sum_{z \in N(x)} \rho(z, x) \tag{3}$$

Suppose there is a mass function defined on $\mathfrak{X}$ by $m : \mathfrak{X} \to R^+$; $m(x) = \#[x] = p$ (total number of points). Then we define the density of a data point as

$$\beta(x) = \frac{1}{\mu(x)} \sum_{x \in N(x)} m(x) \tag{4}$$

Suppose $x \in X$. We use $\pi_i$ to denote the projection operator in the i-th component, i.e., $\pi_i(x) = \pi_i x = x^i$. Respectively, $d_i(x, y) = d(\pi_i x, \pi_i y)$. For $x, y \in \mathfrak{X}$, define $\pi_i x = \{\pi_i z : z \in [x]\}$, $\rho_i(x, y) = \rho(\pi_i x, \pi_i y)$, and $\sigma_i(x) = \sigma(\pi_i x)$. Then we consider projection into the i-th component, the KNN neighbor set N(x) is replaced by $N_i(x) = \{\pi_i x, y_1, \cdots, y_k : \pi_i y_i$ are the k-nearest points to $\pi_i x\}$. The corresponding definition of sparseness and density are

$$\mu_i(x) = \frac{1}{k} \sum_{z \in N_i(x)} \rho_i(z, x), \quad \beta_i(x) = \frac{1}{\mu_i(x)} \sum_{x \in N_i(x)} m(x) \tag{5}$$

Now we describe the process of projected clustering. The main idea is that distance between data objects is restricted to subsets of dimensions where object values are dense [10]. This means that we only consider contributions of relevant dimensions when computing the distance between data point and the cluster center.

Different from [10], we use a fixed threshold value to determine the dense and sparse dimensions. Now let $x \in \mathfrak{X}$ to be a cell data object. Suppose $\beta_0 > 0$ is a positive threshold determined in the process of griding process. Define a matrix $\Theta = [\theta_{ij}]_{N \times n}$ by[1]

$$\theta_{x,j} = \begin{cases} 1, & \text{if } \beta_j(x) \geq \beta_0 \\ 0, & \text{else} \end{cases}; \; j = 1, \cdots, n \tag{6}$$

By this index matrix we obtain a projected cell distance as follows

$$\rho_\pi(x, y) = \left[ \sum_{j=1}^{n} \theta_{x,j} \theta_{y,j} (\rho_j(x, y))^2 \right]^{\frac{1}{2}} \tag{7}$$

## 3. Minimum Spanning Trees

Let G=(V, E) be a connected, undirected edge-weighted graph with N nodes as before. W = [$w_{ij}$] is the weight matrix. For any $P \subseteq V$, we use Q = G-P to denote the subgraph generated by the vertices V\P called a partition of nodes. A spanning subgraph is a subgraph that contains all the vertices of the original graph. A minimal spanning tree of graph G is a spanning graph with no circuits whose weight is minimum among all spanning trees of G.

For a partition P, Q of G, define $\rho(P, Q)$ as the smallest weight among all edges from the cut-set C (P, Q), which is the set of edges connecting P and Q. A link is any edge in C(P, Q) with weight $\rho(P, Q)$. The link set is denoted by $\lambda(P, Q)$ [12].

[1]Notice that the size of this matrix maybe less than *N* since we are using cells instead of data points.

 

There are several ways to build Minimum Spanning Tree (MST) from the graph [11]. Two popular ways to implement the algorithms are the agglomerative and the divisive procedures.

The well-known agglomerative procedures are the Kruskal and Prim's algorithms. The first one works by constructing the tree from initial N isolated vertices of the original graph. All the edges are sorted into a non-decreasing order by their weights. For each edge which is not in the tree, if this edge does not form a cycle with the current tree, then we can add this edge to the tree. In the Prim's algorithm, the tree construction starts with a root node. At each step, among all the edges between the nodes in the tree T and those which are not in the tree yet, the node and the edge associated with the smallest weight to the tree T are added.

The second kind of algorithm is the divisive one called the reverse delete algorithm starting with the full graph. Edges are deleted in order of nonincreasing weights based on the cycle property as long as keeping the connectivity of the graph.

Some well-known properties of MST are summarized in the following theorem [12].

**Theorem 3.1.** The minimum spanning tree T(G) of a graph G has the following properties.

1) T contains at least one edge from $\lambda(P,Q)$ for each partition P, Q.

2) Each edge of T is a link of some partition of G.

3) Let ($C_1$,$C_2$) be a partition of G. If $\rho(P,Q) < \rho(C_1,C_2)$ for each partition (P, Q) of $C_1$, then T ($C_1$) forms a connected subtree of T (G).

Once we have the MST, we can obtain the desired clustering by removing inconsistent edges of MST. The simplest way to define inconsistent edges is using weight measure ratio of the edge with average weight of nearby edges in the tree [12]. If the ratio is larger than a threshold, then it is inconsistent. We can determine a stop criteria by the number of clusters, or a minimum size of any cluster by removing edges which can result in two clusters whose sizes are larger than the minimum cluster size.

If we know the number of clusters k, then clustering can start by removing k-1 arbitrary edges from the tree, creating a k-partition. Then we can minimize the change of the total weight of the current clusters to obtain the final clustering.

To reduce computation complexity, Xiaochun Wang *et al*. proposed a divide and conquer approach [11]. Given a loose estimate of minimum and maximum numbers of data items in each cluster, they propose an iterative approach for MST clustering algorithm in five steps: 1) Start with a spanning tree built by the sequential initialization (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 Divisive Hierarchical Clustering Algorithm (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.

However, if the graph size is not large, we can directly get clustering from the graph. When we use the flexible grids technique to obtain the graph, this is often the case. Anyway, the technique of [11] can be applied to further reduce the computing time.

## 4. Grid Based Spatial Clustering

The grid based clustering uses a multi-resolution grid structure which contains the data objects and acts as operands of clustering performance [1]. For example, the authors [13] propose a gravity based grid which approximates the cell influence by gravity centers. The authors claim that the proposed technique can reduce memory usage and simplify computational complexity with minor loses of the clustering accuracy.

Traditional grids are regular hypercubic grid. This requires the grid construction cover all the data space with the same precision. The second method uses flexible grids, i.e. multi-resolution grids with hypercubic or hyper-rectangular cells having randomly oriented borders [14]. The main clustering technique is a tree based searching with a similarity measure composed of both the density and distance differences [15].

Suppose the data set is $X = [x_1, \cdots, x_N] \subset R^n$. It contains in a rectangle $D_0$ in $R^n$. A grid is a graph G where each node is a complex $v$ = ( *D, norm, c, isCrowded, p*), where *D* is the polyhedra, *norm* is a unit vector indicating normal vector of previous cutting plane, *c* is a point which lies in the grid acting as its center, *isCrowded* is 1 or 0 indicating whether the grid is highly populated or not, and *p* is the number of data points covered by the grid. The initial grid is $D_0$ with an arbitrary normal vector. In each step, we can define the center of the grid as its geometrical center.

For two nodes $v_i = (D_i, norm_i, c_i, isCrowded_i, p_i),\ i = 1, 2$ there exists a connecting edge between them. The weight value on this edge is defined as

$$\rho(D_1, D_2) = \begin{cases} 0, \text{ if } \min\{p_1, p_2\} = 0 \\ \rho_\pi(v_1, v_2), \quad \text{else} \end{cases} \tag{8}$$

 

The graph is constructed in an iterative way. We start with an initial candidate node $v_0 = (D_0, norm_0, c, is\ Crowded, p)$ where $D_0$ is the original rectangle, $norm_0$ is a random selected unit vector, $c$ is the geometrical center of $D_0$, *is Crowded* = 1, and $p$ is the total population number. Then at each step, the cell containing more number of points (controlled by a threshold value $\beta_p$, or larger enough by diameter) controlled by another threshold value $\beta_d$, is split into two subcells by a hyperplane which is orthogonal to the current normal vector. Position of the hyperplane is random. A cell is called crowded if its population is larger than $\beta_p$. Otherwise it is called sparse. If we reach a sparse cell, then add this cell to the node set of the graph. If we reach a cell with diameter less than $\beta_d$, then add this cell to the node set.

This step continues until each cell has a population less than $\beta_p$, or its diameter is smaller than $\beta_d$. Table 1 gives the algorithm for the graph construction process.

By this algorithm, we can generate a hierarchical grid together with a resulting graph. When the graph is generated, the clustering will become grouping nodes of the graph into clusters. A commonly used technique to deal with this problem is the hierarchical clustering [1]. Agglomerative methods iteratively connect vertices that are close to each other with edges. Small clusters are merged with each other building up the hierarchical structure to find the desired larger clustering structure. Divisive methods on the contrary are based on network flows. This is done by iteratively identifying the intercluster edges in a top-down approach.

Once we have completed the graph construction, those nodes in the graph which are not crowded will correspond to vacant area or outliers. Therefore, in order to reduce computing complexity, we first remove all sparse graph nodes with corresponding edges. The resulting graph is G = (V, E) where V is the set of vertices, E the set of weighted edges. An example is shown in Figure 1 with part of its edges.

Now we use $C(X) = \{X_q: q = 1, 2,\ldots k\}$ to denote a clustering of the data set X where O is the set of outliers. Then

$$X = O \bigcup_{C \in C(X)} C \tag{9}$$

**Table 1. Flexible grids construction algorithm**

**Algorithm: Construction of flexible grids**

**Inputs**

$X = \{x_1, \cdots, x_N\}$ dataset of N points in $R^n$.

$D$ : hyper-rectangle containing X

$\beta_p$ : population threshold value.

$\beta_d$ : cell diameter threshold value

**Outputs**

$V = \{v_1, \cdots, v_N\}$ set of vertices

Begin

$V(0) = \{\bar{v}_0 = (D_0, n_0, c_0, 1, p_0)\}$ . Let t = 0.

while $V(t) \neq \varnothing$

for each v = (D, n, c, isCrowded, p) ∈ V(t)

Generate a cutting hyperplane L passing c and with normal vector parallel to n. Cut the current cell v into two subcells $\bar{v}_1, \bar{v}_2$. For each new node, if p < $\beta_p$ or diam(D) < $\beta_d$, add this new cell to the node set V. Else add it to V(t + 1). Let the new norms be orthogonal to n.

end

t + +;

**end**

**End**

 

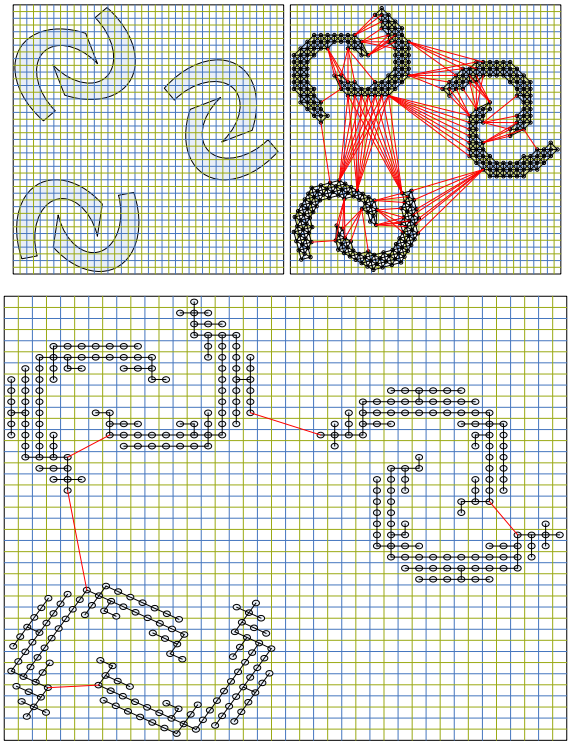

**Figure 1. An example clustering area with three clusters**

For given population threshold value and diameter threshold value, we can generate the flexible grid and obtain a graph. Then we can generate a minimum spanning tree. Then we can get the final clustering. Figure 1 shows an example of minimum spanning tree corresponding to the data set in Figure 1.

Consequently, for fixed $\beta_d$ and $\beta_p$, the data set X is clustered as $C(X,\beta_d,\beta_p)$. We define $|X_q|$ as the number of cells in $X_q$. Define the energy of clustering as the sum of intra-cluster variation measure and inter-cluster distance as

$$E(X,\beta_p,\beta_d) = \frac{1}{k}\sum_{q=1}^{k}\sum_{\substack{v_i\neq v_j \\ v_i,v_j\in X_q}}\frac{1}{|X_q|}\left|\frac{p_i}{diam(D_i)}-\frac{p_j}{diam(D_j)}\right|^2 + \frac{1}{\min\limits_{1\le i\le k}\rho(X_i,\bigcup_{j\neq i}X_j)} \quad (10)$$

By this grid based method, the final clustering can be treated without direct computation to the data points and reduce the number of nodes significantly. The only parameters we need to determine beforehand is the cell crowded threshold value and the minimum cell diameter. One way to choose these two parameters is to optimize the energy of clustering.

## 5. A Performance Study

Suppose that we want to cluster a data set $\Omega\subset R^n$ with N objects. By the graph construction algorithm in the previous section, the data $\Omega$ set is split into a cell hierarchy. A minimum spanning tree is constructed associated with the cell graph.

To make things simpler, we will assume that the cutting planes are perpendicular to one of the axis in this section. Moreover, we assume that each cutting plane passes through the geometrical center of the current cell. Therefore, all the cells are rectangles. Then we can easily count the total number of nodes of the graph.

Suppose the original data cube $D_0$ has a diameter $\sigma_0$, and the initial population threshold is $\beta_p$. For some sufficiently large number M, let $\sigma_i = 2^{-i}\sigma_0, i=1,2,\cdots,M$. Let $\beta_i$ be another decreasing sequence. By choosing appropriate two sequences, we can optimize the clustering.

For specific population and diameter threshold parameters $\beta,\sigma$, let the induced graph be $G(\beta,\sigma)$ with minimum spanning tree $T(\beta,\sigma)$. A clustering of T is denoted by $C(\beta,\sigma)$ is a set of disjoint subtrees whose nodes set coincide with the original tree. We use $\mathfrak{C}(\beta,\sigma)$ to denote the clustering of the data set.

Evidently we have the following properties.

**Theorem 5.1.** Clusters have two properties.

1) Anti-joint property. If $\sigma_1>\sigma_2$, then two disjoint clusters in $\mathfrak{C}(\beta,\sigma_1)$ are disjoint in $\mathfrak{C}(\beta,\sigma_2)$.

2) Monotonicity property. If $\beta_1>\beta_2$, then a cluster in $\mathfrak{C}(\beta_1,\sigma)$ cannot be disintegrated in $\mathfrak{C}(\beta_2,\sigma)$.

Now we assume the population threshold $\beta$ is a constant which determines the granularity of the problem. Therefore we use $\mathfrak{C}(\sigma)$ to denote the clustering. Let us split the energy into two parts, the intra-cluster energy $E_{ia}(\sigma)$ and the intercluster energy $E_{ie}(\sigma)$ as follows

$$E_{ia}(X,\sigma) = \frac{1}{k}\sum_{q=1}^{k}\sum_{\substack{v_i\neq v_j \\ v_i,v_j\in X_q}}\frac{1}{|X_q|}\left|\frac{p_i}{diam(D_i)}-\frac{p_j}{diam(D_j)}\right|^2 \quad (11)$$

$$E_{ie}(X,\sigma) = \frac{1}{\min\limits_{1\le i\le k}\rho(X_i,\bigcup_{j\neq i}X_j)} \quad (12)$$

**Theorem 5.2.** Inter-cluster energy is monotone. That is to say, if $\sigma_1>\sigma_2$, then $E_{ie}(X,\sigma_1)\ge E_{ie}(X,\sigma_2)$.

Proof: It is clear to see that when $\sigma_1>\sigma_2$, a cell maybe split into small cells. Therefore, either the set $X_i$ will be smaller which means that the distance $\rho(X_i,\bigcup_{j\neq i}X_j)$ will become larger.

In a recent work [11] the authors propose a divide and

 

conquer based algorithm consisting of two phases. The first phase includes the sequential initialization and the spanning tree updating, and the second phase uses some technique to locate the longest edges and partitions the obtained approximate minimum spanning tree to form sensible clusters. The authors expect the first phase has $O(fN\log N)$ where $f$ is constant. The average time complexity of the second phase is $O(eN\log N)$ where $e$ is constant. Therefore their expectation of time complexity is $O(N\log N)$. However, our algorithm do provide a time complexity of $O(N\log N)$.

**Theorem 5.3.** Suppose thresholds are $\beta_p = \beta$ and $\beta_d = \sigma$. Then the time complexity of the flexible grids construction algorithm is $O(N\log N)$.

Proof: At each stage, if a cell $v = (D, n, c, isCrowded, p)$ is sparse, i.e., *isCrowded* =0, then the cell is a node in the graph. Otherwise, a cutting hyperplane L passing c and with normal vector parallel to n is generated. The current cell v is cut into two subcells $\overline{v}_1, \overline{v}_2$. For each new node, if $p < \beta_p$ or $diam(D) < \beta_d$, add this new cell to the node set V.

In this process, the computation of *isCrowded* and *p* both have a time of $O(N_i)$ where $N_i$ is the number of data points in the new cell. Ideally, the plane *L* passes the center of the cell *v*. Hence $N_i = p/2$ for i = 1; 2. If not so, the plane is randomly placed by a uniform distribution which we expect the same property. Let the total time complexity be T(N). Then we have T(N) = 2T(N=2)+O(N). Hence we know that T(N) =O(N log N).

## 6. Conclusions

In this paper we present a new projection clustering technique based on grid architecture and minimum spanning tree. The effective using of minimum spanning tree can possibly reduce computing complexity although the construction of the graph and the tree are relatively complicated. The main ingredient here is the application of grid clustering to projection clustering.

Apparently, this research will lead to efficient algorithms. In future work, we will give experimental study on the new technique. This will be lengthy, for the clustering is essentially an optimization process. The best population threshold $\beta_p$ is to be determined which optimizes the clustering energy presented in Section 3. For this reason, we will present this part of the research in another paper, together with an application.

## 7. Acknowledgments

This project is carried out under the "Taishan Scholar" project of Shandong, China.

Research is also supported by the Natural Science Foundation of China (No.60873058), the Natural Science Foundation of Shandong Province (No. Z2007G03), and the Science and Technology Project of Shandong Education Bureau.

## REFERENCES

[1] J. W. Han and M. Kamber, "Data mining concepts and techniques," 2nd Edition, Elsevier, Singapore, 2006.

[2] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in Proceedings 2nd International Conference on Knowledge Discovery and Data Mining, Portland, OR, pp. 226–231, 1996.

[3] R. Bar-Or and C. van Woudenberg, "A novel gravity-based clustering method, technical report," Department of Applied Mathematics, University of Colorado, Denver, CO 80202, 2001.

[4] R. Bar-Or and C. van Woudenberg, "Gene expression analysis with a novel gravity-based clustering method," pp.1–46, December 2001.

[5] R. T. Ng and J. Han, "Clarans: A method for clustering objects for spatial data mining," IEEE Trans. on Knowledge and Data Engineering, Vol. 14, No. 5, pp. 1003–1016, September/October 2002.

[6] M. Li, M. K. Ng, Y. M. Cheung, and J. Huang, "Agglomerative fuzzy K-Means clustering algorithm with selection of number of clusters," IEEE Trans on Knowledge and Data Engineering, Vol. 20, No. 11, pp. 1519–1534, 2008.

[7] Z. He, X. Xu, and S. Deng, "Scalable algorithms for clustering large datasets with mixed type attributes," International Journal of Intelligent Systems, Vol. 20, pp. 1077–1089, 2005.

[8] Z. F. He and F. L. Xiong, "A constrained partition model and K-Means algorithm," Journal of Software, Vol.16, No.5, pp. 799–809, 2005.

[9] Z. Y. Zhang and H. Y. Zha, "Principal manifolds and nonlinear dimensionality reduction via tangent space alignment," SIAM Journal of Scientific Computing, Vol. 26, No. 1, pp. 313–338, 2004.

[10] M. Bouguessa and S. R. Wang, "Mining projected clusters in high-dimensional spaces," IEEE Transaction on Knowledge and Data Engineering, Vol. 21, No. 4, pp. 507–522, 2009.

[11] X. C. Wang, X. L. Wang, and D. M. Wilkes, "A divide-and-conquer approach for minimum spanning tree-based clustering," IEEE Trans on Knowledge and Data Engineering, Vol. 21, No. 7, pp. 945–958, 2009.

[12] C. T. Zahn, "Graph-theoretical methods for detecting and describing gestalt clusters," IEEE Trans. Computers, Vol. 20, No. 1, pp. 68–86, January 1971.

[13] C. H. Li and Z. H. Sun, "A mean approximation approach to a class of grid-based clustering algorithms," Journal of Software, Vol. 14, No. 7, pp. 1267–1274, 2003.

[14] Marc-Ismael, Akodj`enou-Jeannin, Kav´e Salamatian, and P. Gallinari, "Flexible grid-based clustering," J. N. Kok *et al.* (Eds.), LNAI 4702, PKDD, pp. 350–357, 2007.

[15] U. Brandes, M. Gaertler, and D. Wagner, "Experiments on graph clustering algorithms," In ESA, pp. 568–579, 2003.

 