Volume 4, Issue 4, April 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Commonly Used Spatial Data Mining Data Structures – A Survey

G. Dona Rashmi *
Assistant Professor, Department of MCA
Karpagam College of Engineering, India

Dr.V. Narayani
Associate Professor, Department of MCA
Karpagam College of Engineering, India

Abstract - This paper deals with spatial data structures for indexing and with their usability for knowledge discovery in spatial data. Huge amount of data processed in spatial data mining requires using some indexing structures to speed up the mining process. Typical data types and operations used in geographic information systems are described in this paper. Then basic spatial data mining tasks and some spatial data mining systems are introduced. Finally, indexing spatial structures for both vector and metric spaces are described and structures used in some spatial data mining systems are presented.

Keywords: Geographical Information System (GIS), Spatial Data Mining, Geo Miner, GWiM

I. SPATIAL DATA TYPES IN GIS

A geographic information system is a special kind of information system, which allows manipulate, analyze, summarize, query, edit and visualize geographically related data. Geographically related data are composed of:

- spatial attributes, e. g. coordinates, geometry
- Non-spatial attributes, e. g. name of town, number of inhabitants

Spatial data in GIS may be represented in raster or vector model. *Raster model* divides space into a regular grid of cells, usually called pixels. Each cell contains a single value and its position is defined by its indices in the grid. The resolution of the raster depends on its pixel size [1]. The smaller the pixel size, the higher the resolution, but also the larger the data size.

Vector model represents spatial objects with data structures, whose basic primitive is a point. This allows precise representation of coordinates and it is useful for analysis [8]. Data structures used to store spatial objects in the vector model are:

- *point* – defined by its coordinates
- *chain* – sequence of points connected with lines
- *polygon* – sequence of connected chains (the last point of one chain is the first point of the other chain), it is enclosed and the chains must not intersect

Fundamental operations used to manipulate vector data are:

- determining the distance of two objects
- determining the area of the object (if it is a polygon)
- determining the length of the object (if it is a chain or polygon)
- determining an intersection or union of the objects
- determining a mutual position of two objects (they can intersect, overlap, touch, one can contain the other, . . .)

II. SPATIAL DATA MINING

Analysis is an important part of GIS which allows spatial operations with data (e. g. network analysis or filtering of raster data), measuring functions (e.g. distance, direction between objects), statistic analyses or terrain model analysis (e. g. visibility analysis).

Spatial data mining is a special kind of data mining [1]. The main difference between data mining and spatial data mining is that in spatial data mining tasks we use not only non-spatial attributes (as it is usual in data mining in non-spatial data), but also spatial attributes.

A. Spatial data mining tasks

Basic tasks of spatial data mining are:

- *Classification* – finds a set of rules which determine the class of the classified object according to its attributes. E.g."IF population of city = high AND economic power of city = high THEN unemployment of city = low" or classification of a pixel into one of classes, e. g. water, field, forest.

- *Association rules* – find (spatially related) rules from the database. Association rules describe patterns, which are often in the database.

The association rule has the following form:
A ! B(s%; c%), where s is the support of the rule (the probability, that A and B hold together in all the possible cases) and c is the confidence (the conditional probability that B is true under the condition of A[2] e. g. "if the city is large, it is near the river (with probability 80%)" or "if the neighboring pixels are classified as water, then central pixel is water (probability 80%)."

- *Characteristic rules* – describe some part of database. E.g."bridge is an object in the place where a road crosses a river."
- *Discriminate rules* – describe differences between two parts of database. e. g. find differences between cities with high and low unemployment rate.
- *Clustering* – groups the object from database into clusters in such a way that object in one cluster are similar and objects from different clusters are dissimilar. e. g. we can find clusters of cities with similar level of unemployment or we can cluster pixels into similarity classes based on spectral characteristics.
- *Trend detection* – finds trends in database. A trend is a temporal pattern in some time series data.

A spatial trend is defined as a pattern of change of a non-spatial attribute in the neighborhood of a spatial object [12]. E.g." when moving away from Brno, the unemployment rate increases" or we can find changes of pixel classification of a given area in the last five years.

B. Spatial data mining systems

GeoMiner

The GeoMiner is a system for knowledge discovery in large spatial databases. It was developed at Simon Fraser University in Canada. The GeoMiner is an extension of and developed from DBMiner. The DBMiner is a relational data mining system, which uses Microsoft SQL Server 7.0 to store data [12].
It contains the five following data mining modules:
Association, *Classification*, *Clustering*, *3D Cube Explorer* (displays data cube in a 3D view) and *OLAP Browser* (generalizes data in a spreadsheet or graphical form).

The Geominer is composed of the following modules: *Geo–characterizer*, *Geo–associator*, *Geo–cluster analyzer*, *Geo–classifier* (their function is similar to the previous definition) and *Geo–comparator* (its function corresponds to the discriminant rules in the previous definition).

Descartes

System Descartes supports the visual analysis of spatially referenced data. It uses two basic tools: automatic visualization and interactive manipulation with maps [7]. The system uses the following methods to visualize information: *area coloration*, *charts* and combination of both of them. The area coloration represents a numeric attribute as color: the greater the value, the darker the color of the region. Descartes offers various types of charts (bar, pie, etc.).

Fuzzy Spatial OQL for Fuzzy KDD

Language is designed to select, process and mine data from Spatial Object–Oriented Databases and it is based on a fuzzy set theory. In the classical theory, given a set S and an element e, we can decide whether this element belongs or does not belong to S. In the fuzzy set theory, the probability that e belongs to S can vary from 0 to 1.

In the fuzzy spatial OQL, fuzzy values are used in the where clause of a fuzzy query and the answer is a fuzzy set of elements defined by these fuzzy values. In the fuzzy decision tree, each node is associated with a test on the values of some attribute [7]. All the edges of the node are labeled by fuzzy values. This enhances the comprehensibility of the decision tree.

GWiM

GWiM is a system for knowledge discovery in spatial data. It is built upon the WiM system. WiM uses inductive logic programming to synthesize closed Horn clauses. Query language of GWiM contains three types of queries. Two of them, characteristic and discriminant rules, are adaptation of query language of DBMiner [9]. The dependency rules describe a dependency between different classes.
extract <KindOfRule> rule
for <NameOfTarget>
[from <ListOfClasses>] [<Constraints>]
[from point of view <ExplicitDomainKnowledge>];

The clause <KindOfRule> determines which rule we want to mine (classification, characteristic or discrimination). Clause <NameOfTarget> determines the object we want to discover the knowledge about. <ListOfClasses> contains a list of tables where data for mining are stored. Data from these tables are selected according to the condition <condition>, which is similar to the languageSQL. Clause <DomainKnowledge> contains some other information necessary for knowledge discovery.

III. SPATIAL DATA STRUCTURES IN GIS

A. Quad tree

The quad tree is used to index 2D space. Each internal node of the tree splits the space into four disjunct subspaces (called NW, NE, SW, SE) according to the axes [11]. Each of these subspaces is split recursively until there is at most one object inside each of them

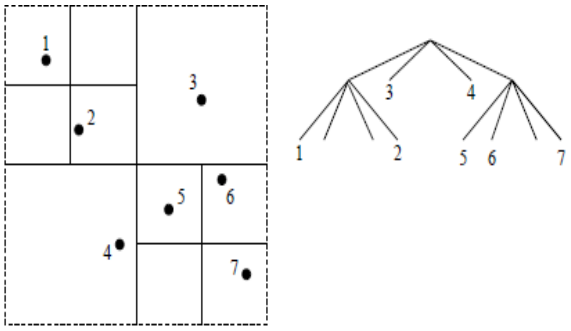

Fig.1 Quad Tree

The quad tree is not balanced and its balance depends on the data distribution and the order of inserting the points.

B. k–d–tree

This method uses a binary tree to split k–dimensional space. This tree splits the space into two subspaces according to one of the coordinates of the splitting point. Let level (node) be the length of the path from the root to the node nod and suppose the axes are numbered from 0 to k − 1. At the level level(nod) in every node the space is split according to the coordinate number (level(nod) mod k).

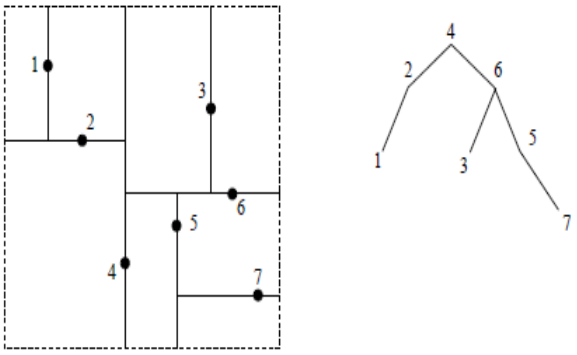

Fig. 2 K-d Tree

Inserting and searching are similar to the binary trees. We only have to compare nodes according to the coordinate number (level(nod) mod k). This structure has one disadvantage: it is sensitive to the order in which the objects are inserted.

C. R–tree

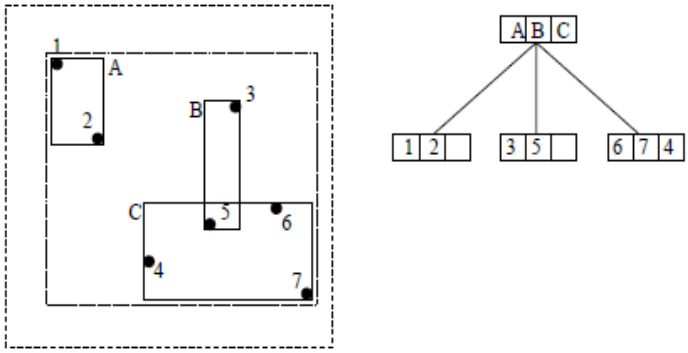

Fig. 3 R- Tree

The R–tree is the modification of B-tree for spatial data. This tree is balanced and splits the space into the rectangles which can overlap [3]. Each node except root contains from m to M children, where 2 <= m <= M/2. The root contains at least 2 children unless it is a leaf. Figure 3 shows an example of r-tree of the order 3.

The node is represented by the minimum bounding rectangle containing all the objects of its subtree. Each children of the node is split recursively. Pointers to the data objects are stored in the leafs.

D. R*–tree

R*-tree is a modification of R–tree which uses different heuristic for operation INSERT. R–tree tries to minimize the area of all nodes of the tree. R*–tree combines more criteria: the area covered by a bounding rectangle, the margin of a rectangle and the overlap between rectangles. The implementation of this method is harder, but R*–trees are more effective than R-trees.

E. R+–tree

R+–tree is an extension of the R–tree. In contrast to R–tree bounding rectangles of the nodes at one level don't overlap in this structure. This feature decreases the number of searched branches of the tree and reduces the time consumption. In the R+–tree it is allowed to split data objects so that different parts of one object can be stored in more nodes of one tree level (figure 4). If a rectangle overlaps another one, we decompose it into a group of nonoverlaping rectangles which cover the same data objects. This increases space consumption but allows zero overlap of the nodes and therefore reduces the time consumption.

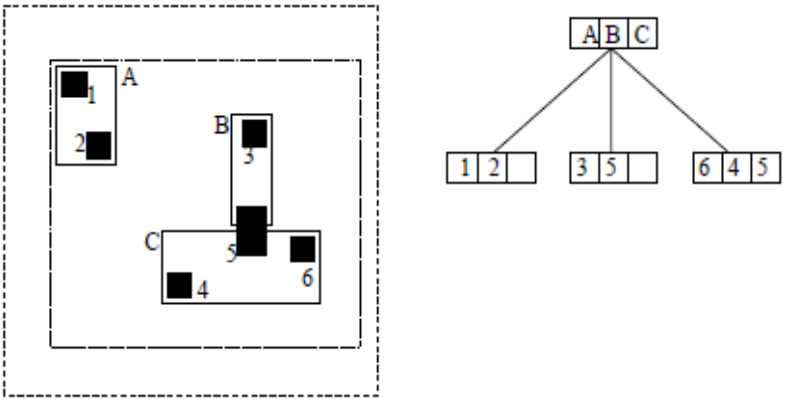

Fig.4 R+ Tree

IV. DATA STRUCTURES FOR METRIC SPACES

A. Vp–tree

The vp–tree (or vantage point tree) partitions the data space around selected points and forms a hierarchical tree structure (Figure 5). These selected points are called **vantage points**. Each internal node of the tree is of the form (PV; M; R; L) where:

- PV is the vantage point
- M is the median distance among the distances of all the points (belonging to the subtree of the node) from PV
- R, L are the pointers to the sons of the node. Left (right) son contains the points whose distances from PV are less than or equal (greater than or equal) to M.

The leafs contain pointers to the data points. The vp-tree is used to find the objects whose distance from Q is less than or equal to r:

1. If d(Q; PV) _ r, then PV is in the answer set.
2. If d(Q; PV) + r _ M, then recursively search the right subtree.
3. If d(Q; PV) − r _ M, then recursively search the left subtree.

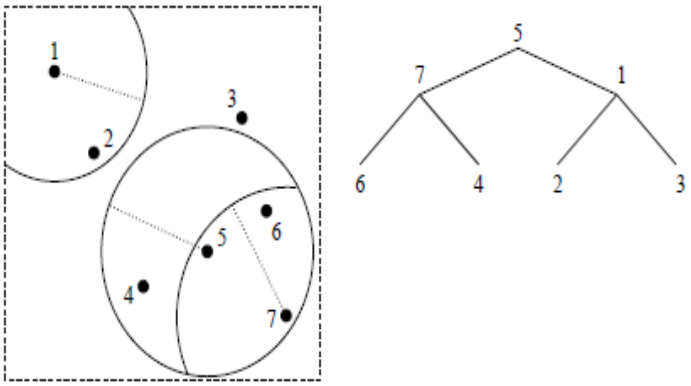

Fig.5 Vp-Tree

B. M–way vp–tree

M–way vp–tree (or multi–way vp–tree) [5] is one of the modifications of the vp-tree which decreases the height of the tree. The structure of m–way vp-tree of order m is very similar to the vp-tree. The main difference is that it splits objects into m groups according to their distances from the vantage point. The splitting values, called *cutoff* values, are

stored in a node. Figure 6 shows an example of m-way vp-tree of order 3. The construction of the tree requires O(nlogmn) distance computations. That is log2m times better than binary vp-trees.

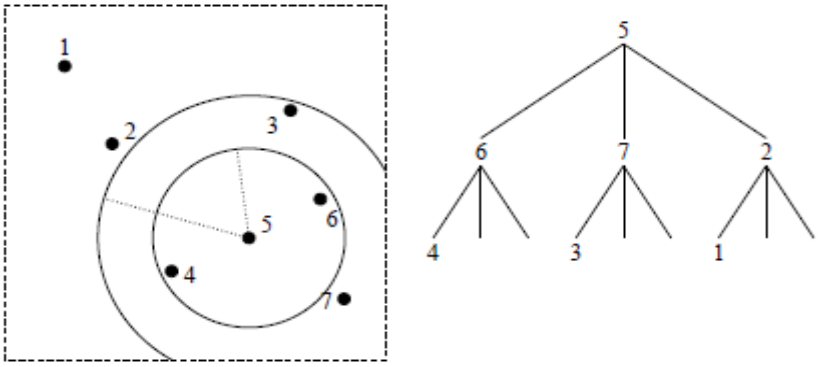

Fig.6 M-Way Vp-Tree

M–way vp–tree has one disadvantage if it partitions high–dimensional spaces. In this case, spherical cuts of one node are very thin and search operation will have to search more than one branch very often.

C. Multi-vantage-point tree

Another modification of the vp-tree, *mvp-tree* (or multi–vantage–point tree), uses two vantage points (PV 1 and PV 2) to partition data in one node. Each internal node of the tree can be seen as two levels of vp-tree (see Figure 7). At the first level it partitions data according to PV 1 and at the second level it partitions all the children of PV 1 according to PV 2[6]. Using the same vantage point for all the children saves the space. All the splitting values for both levels are stored in the node. If the vp-tree in the node is of order m, the node splits the space into m2 partitions. In a leaf node, data points and their distances from both vantage points are stored. Moreover, leaf nodes contain extra information about their distance from the vantage points of the first p nodes in the path from the root to the leaf node. This information is used to reduce the number of distance computations during the search operation. The construction of the tree requires O(nlogmn) distance computations.

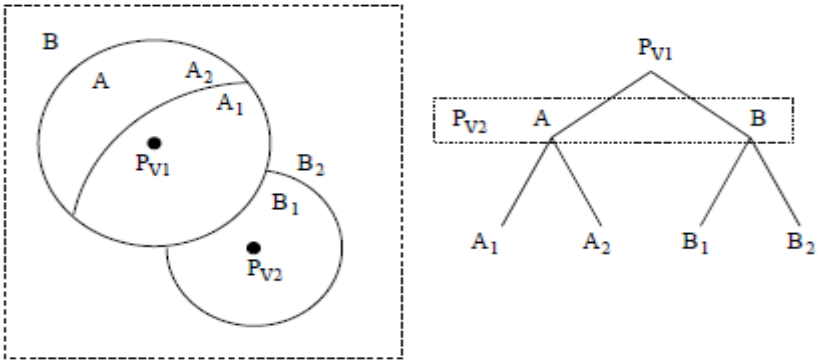

Fig.7 One Node of mvp -Tree

D. M–tree

The M–tree is designed to partition a *metric space* with a distance function d (but it works for vector spaces, too). This function has the following properties:

1. symmetry: d(x; y) = d(y; x)
2. non negativity: d(x; y) > 0 if x 6= y; d(x; x) = 0
3. triangle inequality: d(x; y) _ d(x; z) + d(z; y)

The M–tree partitions objects according to their relative distance. The distance function d depends on the concrete application. The goal of M–tree is to reduce not only the number of accessed nodes during the search, but also the number of distance computations, because this operation can be very expensive[9]. Nodes of the tree have a fixed size. Indexed objects (or pointers to them) are stored in the leafs. For each entry of the leaf the distance to the Parent is also stored.

Each entry of the internal node has the following structure:

- routing object Or
- pointer to the subtree ptr
- covering radius r
- distance of Or from its parent

Routing objects are used to partition the space: all the objects in the subtree of Or (referenced by ptr) are within the distance r (r > 0) from Or. Distance of Or from its parent is the distance from the object which references the node where the Or is stored. This information is used to decrease the number of distance computations during the SEARCH operation. Figure 8 shows an example of m-tree of order 3.

The structure is primarily designed for similarity search in metric spaces. So it could be used for similarity search or nearest neighbor search in spatial data mining too. This structure could be useful e. g. for efficient clusterization in spatial data mining.

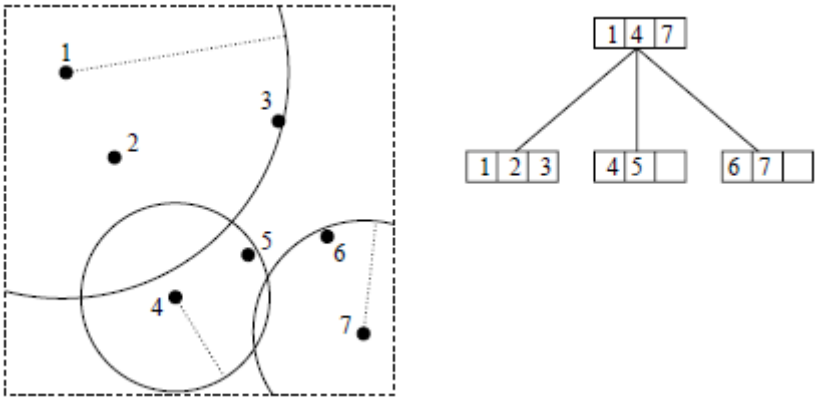

Fig.8 m-Tree

V. DATA STRUCTURES IN SPATIAL DATA MINING

A. Neighborhood graphs Neighborhood graphs and neighborhood paths

Definition: *Neighborhood graph* G for spatial relation neighbor/2 is a graph G(U,H) where U is a set of nodes and H is a set of edges[7]. Each node represents an object and two nodes N1, N2 are connected by edge iff the objects corresponding to N1 and N2 are in the relation neighbor [4]. The relation neighbor can be:

- *topological relation*, e. g. two objects touch, cover, are equal, contain
- *metric relation*, e. g. distance of the objects is less than d
- direction relation, e.g. north, south, east, west
- any conjunction or disjunction of previous relations

Neighborhood graph is oriented. Thus it can happen that object A is a neighbor of the object B but object B is not a neighbor of the object A.

Definition: *Neighborhood path* for the neighborhood graph G is an ordered list of nodes from G where every two following nodes from the path are connected by some edge from G, i. e. for the path [n0, n1, : : :, nk−1] there must be edges (ni, ni+1) for every 0 <= i < k − 1. *Length of the path* is a sum of edges in the path.

Elementary operations on the neighborhood graphs

Elementary operations on the neighborhood graphs are:

get Graph(data, neighbor) – returns the neighborhood graph G representing the relation neighbor on the objects from the table data[8]. The relation neighbor can be one of the spatial relations listed in the definition of the neighborhood graph.

get Neighborhood(G, o, pred) – returns the set of the objects connected to the object o by some of the edges from the graph G. The predicate pred must hold for these objects. This condition is used if we want to get only some specific neighbors of the object o. The predicate pred may not necessarily be spatial.

create Path(G, pred, i) – returns the set of all paths which consist of the nodes and edges from the graph G, their length is less than or equal to i and the predicate pred holds for them[1]. Moreover these paths must not contain any cycles, i. e. every node from G can appear at most once in each path.

Neighborhood graphs in spatial data mining

In neighborhood graphs are used to represent topology of data objects and their neighborhood. Four spatial data mining tasks that use neighborhood graphs are described: spatial association rules, spatial clustering, spatial trend detection and spatial classification. In GeoKD system spatial data are stored in structures, points, chains and polygons. Program uses operation get Graph from previous definition to create neighborhood graph from these data[5]. The graph is stored in database and is used to find neighbors of an object during the knowledge discovery process.

B. Data structures in GWiM

System GWiM uses prolog facts to store learning data. For example prolog fact for object city with attributes name Brno, population 384 369, unemployment 7% can be: city("Brno", 384369, 7). Main disadvantage of this system is a problem with processing large amount of data. It doesn't use any index method to access data. In partial solution of this problem was proposed, but it wasn't implemented.

C. Mining in raster data – satellite images interpretation

The raster image interpretation is an important method of GIS analysis. Given a multispectral satellite image, the goal is to classify all pixels according to their land cover types (e. g. water, field, forest).

The classification of pixel (i. e. land cover type of pixel) depends not only on its own characteristics, but also on the characteristics of neighboring pixels.

VI. CONCLUSION

Spatial data mining is the process of discovering interesting and previously un-known, but potentially useful patterns from large spatial datasets. In this paper an overview of indexing spatial structures for both vector and metric spaces are described and structures used in some spatial data mining systems are also discussed. This paper helps to identify what are all the data structures used in spatial databases. In future these data structures are implemented in different domains.

REFERENCES

[1] Andrienko G., Andrienko N.: Data Mining with C4.5 and Cartographic Visualization. N.W.Paton and T.Griffiths (eds.) User Interfaces to Data Intensive Systems 1999, IEEE Computer Society Los Alamitos, CA, pp. 162-165. ISBN 0-7695-0262-8.
[2] Data Mining and Knowledge Discovery:http://www.digimine.com/usama/datamine/
[3] Bentley J.: Multidimensional Binary Search Trees used for Associative Searching. Communications of ACM, 18, 9, Sep. 1975, pp. 509-517.
[4] J. Han, K. Koperski, and N. Stefanovic: GeoMiner: A System Prototype for Spatial Data Mining. Proc. 1997 ACM-SIGMOD Int'l Conf. on Management of Data(SIGMOD'97), Tucson, Arizona, May1997 (System prototype demonstration).
[5] Tu˘cek J.: Geografick´e informa˘cn´ı syst´emy. Principy a praxe. Computer Press 1998.
[6] Paolo Ciaccia, Marco Patella, Pavel Zezula: M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. VLDB 1997: 426-435.
[7] GeoMiner: http://db.cs.sfu.ca/GeoMiner/
[8] GeoInformatica: http://www.wkap.nl/journalhome.htm/1384-6175
[9] Hor´ak J.: *Anal´yzy dopravn´ı dostupnosti a obslu˘znosti* Proc. of Conf. GIS... Ostrava 2001.
[10] Gaede V., G¨ unther O.: *Multidimensional Access Methods.* ACM Computing Surveys, Vol. 30, No. 2, June 1998, pp. 170 - 231.
[11] Finkel R., Bentley J.: *Quad Trees: A Data Structure for Retrieval on Multiple Keys.* Acta Informatica, Vol. 4, No. 1, 1974, pp. 1-9.
[12] DBMiner: http://www.dbminer.com

