FI MU

Faculty of Informatics
Masaryk University

Data Structures for Spatial Data Mining

by

Petr Kuba

FI MU Report Series FIMU-RS-2001-05

September 2001

Data structures for spatial data mining

Petr Kuba
Department of Computer Science, Faculty of Informatics
Masaryk University Brno, Czech Republic
xkuba@fi.muni.cz

September 4, 2001

Abstract

This report 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 (or in data mining generally) 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.

1 Spatial data types in GIS

A geographic information system [43] is a special kind of information system, which allows manipulate, analyse, 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. 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's useful for analysis. 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, ...)

2 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 [11, 24] is a special kind of data mining [12]. 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 datamining in non-spatial data), but also spatial attributes.

2.1 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: $A \rightarrow B(s\%, c\%)$, 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 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."

- *discriminant 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 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.

In the rest of this section, we will describe several existing systems which can be used for spatial data mining and information sources concerning spatial data mining.

2.2 Spatial data mining systems

GeoMiner

The GeoMiner [17, 21] 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 [9, 20]. The DBMiner is a relational data mining system, which uses Microsoft SQL Server 7.0 to store data. 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).

The system contains its own language for knowledge discovery in spatial data and it uses graphical interface to communicate with the user and display results in the form of graphs, charts, maps, etc.

Descartes

System Descartes [8] supports the visual analysis of spatially referenced data. It uses two basic tools: automatic visualization (presentation of the data on the map) and interactive manipulation with maps. 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.). The combination of both the methods allows to visualize one attribute with area coloration and another attribute (or attributes) with charts.

In [1] the integration of Descartes with Kepler is used to classify spatial related data. Kepler [44] is a knowledge discovery system with a plug–

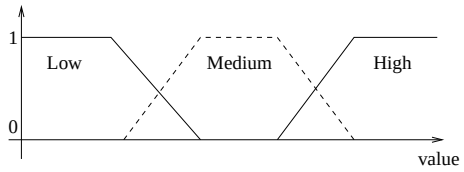

Figure 1: Fuzzy values for numerical attribute

in architecture. The classification system C4.5 [38] is one of the pluggins. In this integrated system, Kepler is used to classify data and Descartes is used to visualize and analyse source data and results of the classification on the map.

Fuzzy Spatial OQL for Fuzzy KDD

In [4], a fuzzy spatial object query language and fuzzy decision tree [30] are introduced. This 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 figure 1, a numerical attribute $value$ can have the fuzzy values of Low, $Medium$ and $High$.

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. All the edges of the node are labeled by fuzzy values. This enhances the comprehensibility of the decision tree.

GWiM

In [35], the use of Inductive Logic Programing for knowledge discovery in spatial data is discussed and an inductive query language is proposed. Then, a description of mining system GWiM is given.

GWiM is a system for knowledge discovery in spatial data. It is built upon the WiM system. WiM [14] 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. The Following clauses `<NameOfTarget>`, `<ListOfClasses>` and `<Constraints>` specify the data which should be used for mining. The clause `<ExplicitDomainKnowledge>` is a list of predicates or hierarchy of predicates. The answer to these queries is a formula of first–order logic which characterizes the subset of the database specified by the rule.

GeoKD

In [26, 27], a language for knowledge discovery in spatial data is proposed. Interpreter of this language is called GeoKD. This language contains three kinds of rules (classification, characteristic and discrimination rules) and it uses neighborhood graphs to represent spatial data. Syntax of the language is very similar to GWiM and GeoMiner:

```
extract <KindOfRule> rule
for <NameOfTarget>
from <ListOfClasses>
where <condition>
from point of view <DomainKnowledge>;
```

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 language SQL. Clause `<DomainKnowledge>` contains some other information necessary for knowledge discovery. For example, it contains information where spatial data for this query are stored.

Source data for interpreter of this language are stored in the database PostgreSQL [36]. Interpreter uses the three programs for data mining: C4.5 [38], Rt4.0 [39] and Progol [37]. C4.5 and Rt4.0 are used to find classification rules and Progol is used to find characteristic and discrimination rules. Interpreter unifies access to these systems and allows to use them for spatial datamining.

2.3 Information sources

Papers from the area of knowledge discovery in spatial data can be found in conference proceedings and journals that focus on GIS or knowledge discovery in databases. Here are some basic of them: *European Conference on Principles of Data Mining and Knowledge Discovery (PKDD)* [33], *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *ACM International Workshop on Advances in Geographic Information Systems* [25], *International Conference on Geographic Information Science (GIScience) [18]* and journals *Data Mining and Knowledge Discovery* [7] and *GeoInformatica* [16].

Local conferences in the Czech republic are *GIS... Ostrava*, *Dobývání znalostí z databází* or *Znalosti*.

Another useful information source is the project *Spin!* [41] where new Spatial Data Mining system is being developed, or *The National Center for Geographic Information and Analysis* (NCGIA) [31].

The rest of the report is organized as follows. In section 3 basic spatial data structures used in GIS are presented. In section 4, spatial structures designed primarily for metric spaces are described. And finally, structures used in some spatial data mining systems are presented in section 5.

3 Spatial data structures in GIS

3.1 Quad tree

The quad tree [13, 34] 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. Each of these subspaces is split recursively until there is at most one object inside each of them (see Figure 2).

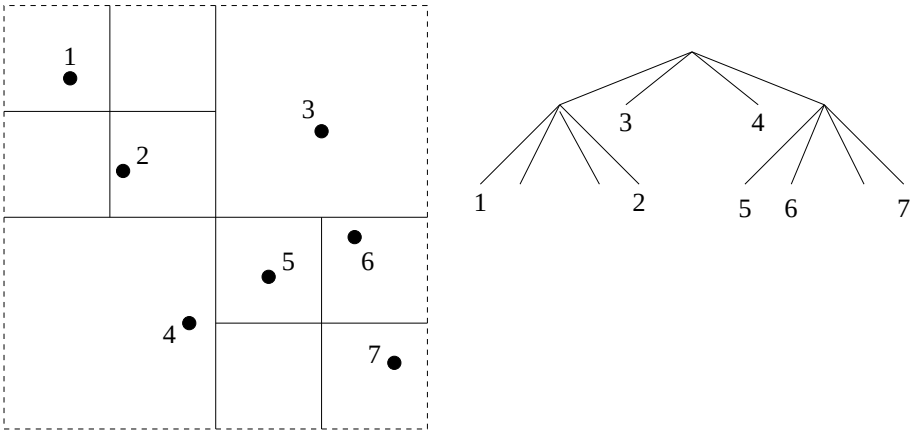

Figure 2: Quad tree

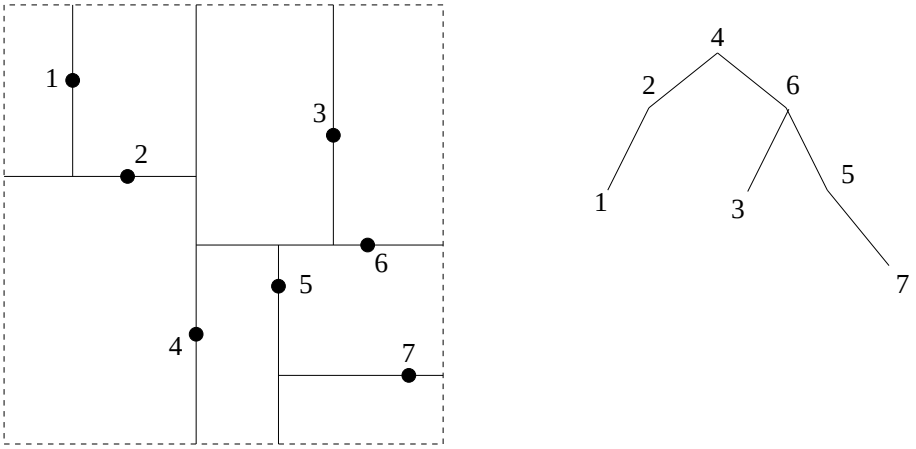

Figure 3: k–d–tree

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

3.2 k–d–tree

This method uses a binary tree to split k–dimensional space [3, 15, 34]. This tree splits the space into two subspaces according to one of the coordinates of the splitting point (see Figure 3).

Let $level(nod)$ 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 num-

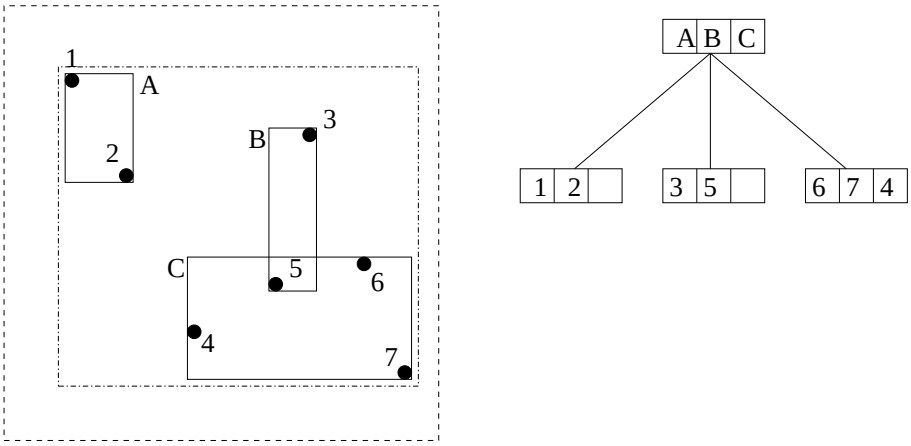

Figure 4: R–tree

ber $(level(nod)\, mod\, k)$.

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.

3.3 R–tree

The R–tree [19, 34] is the modification of B-tree for spatial data. This tree is balanced and splits the space into the rectangles which can overlap. Each node except root contains from m to M children, where $2 \leq m \leq M/2$. The root contains at least 2 children unless it is a leaf. Figure 4 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 of children of the node is split recursively. Pointers to the data objects are stored in the leafs.

Because of the overlapping of bounding rectangles it could be necessary to search more than one branch of the tree. Therefore, it is important to separate the rectangles as much as possible. This problem must be solved by the operation INSERT which uses some kind of heuristic. It finds such a leaf that inserting a new object into it will cause as small changes in the tree as possible.

The splitting operation is also important. We want to decrease the probability that we will have to search both new nodes. Testing all the pos-

sibilities has exponential complexity, so the algorithms for approximate solution are used.

The R-tree is one of the most cited spatial data structures and it is very often used for comparison with new structures.

3.4 R*–tree

R*-tree [2, 34] 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 goal of decreasing the area covered by a bounding rectangle is to decrease the dead space, it means the space covered by the bounding rectangle but not by the enclosed rectangles. This decreases the number of branches that are searched uselessly. Minimization of the margin (the sum of the lengths of the edges) of a bounding rectangle prefers the squares. Minimization of the overlap between rectangles decreases the number of paths that must be searched.

The implementation of this method is harder, but R*–trees are more effective than R-trees.

3.5 R+–tree

R+–tree [40] 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 (see figure 5). If a rectangle overlaps another one, we decompose it into a group of nonoverlaping rectangles which cover the same data objects. This increases a space consumption but allows zero overlap of the nodes and therefore reduces the time consumption.

In this section we have discussed data structures that are used to partition vector spaces. In the following section we describe structures that are primarily used to partition metric spaces.

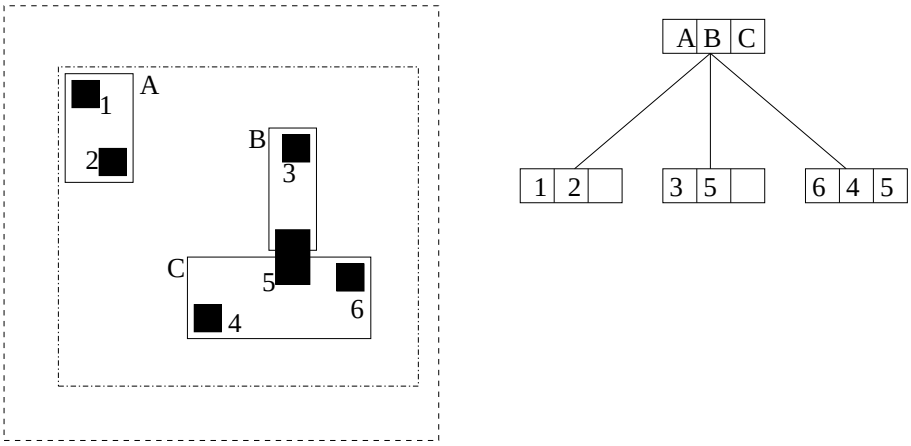

Figure 5: R+–tree

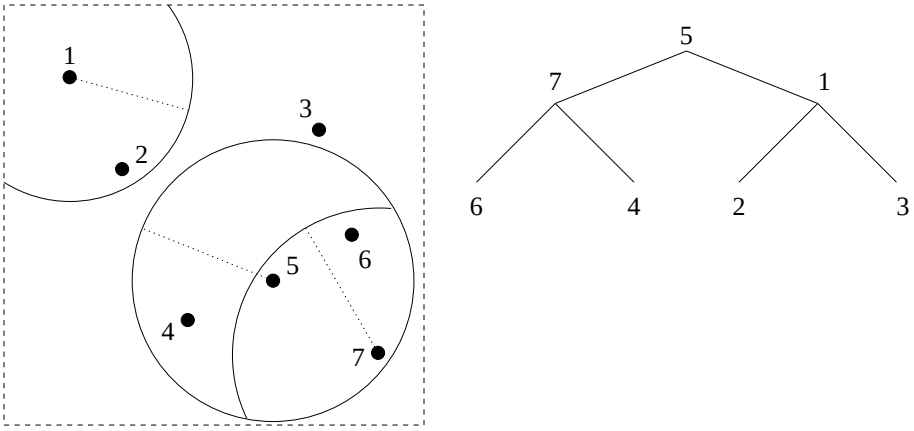

Figure 6: vp–tree

4 Data structures for metric spaces

4.1 Vp–tree

The vp–tree (or vantage point tree) [5] partitions the data space around selected points and forms a hierarchical tree structure (see Figure 6). These selected points are called **vantage points**.

Each internal node of the tree is of the form (P_V, M, R, L) where:

- P_V is the vantage point
- M is the median distance among the distances of all the points (be-

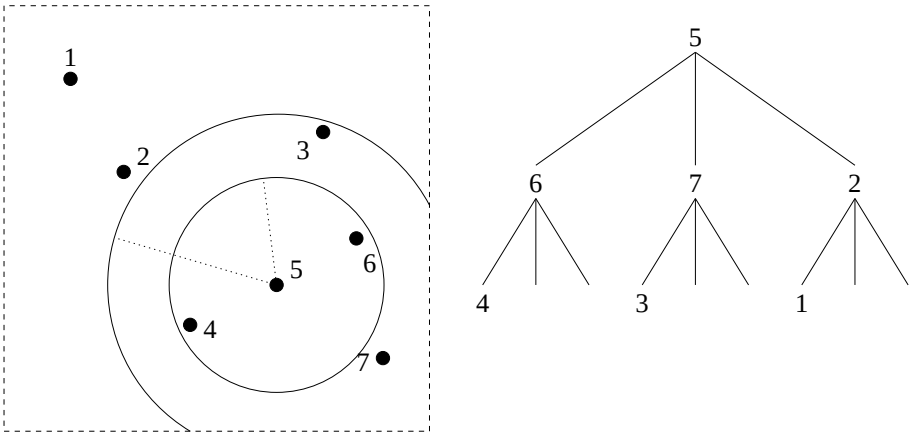

Figure 7: M–way vp–tree

longing to the subtree of the node) from P_V

- R, L are the pointers to the sons of the node. Left (right) son contains the points whose distances from P_V 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 $d(Q, P_V) \leq r$, then P_V is in the answer set.
2. If $d(Q, P_V) + r \geq M$, then recursively search the right subtree.
3. If $d(Q, P_V) - r \leq M$, then recursively search the left subtree.

4.2 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 7 shows an example of m-way vp-tree of order 3.

The construction of the tree requires $O(nlog_m n)$ distance computations. That is $log_2 m$ times better than binary vp-trees.

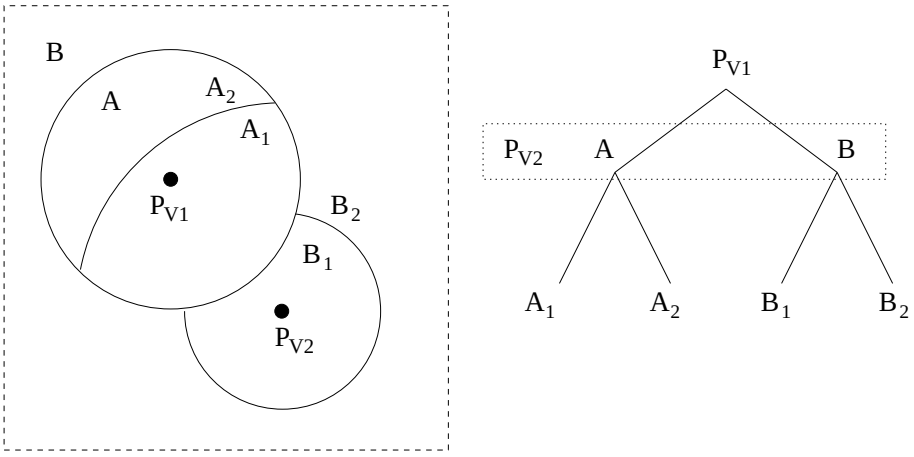

Figure 8: One node of mvp–tree

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.

4.3 Multi-vantage-point tree

Another modification of the vp-tree, *mvp-tree* (or multi–vantage–point tree) [5], uses two vantage points (P_{V1} and P_{V2}) to partition data in one node. Each internal node of the tree can be seen as two levels of vp-tree (see Figure 8). At the first level it partitions data according to P_{V1} and at the second level it partitions all the children of P_{V1} according to P_{V2}. 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 m^2 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 $O(nlog_m n)$ distance computations.

4.4 M–tree

The M–tree [6] 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 $x \neq y$; $d(x, x) = 0$
3. triangle inequality: $d(x, y) \leq d(x, z) + d(z, y)$

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.

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 O_r
- pointer to the subtree ptr
- covering radius r
- distance of O_r from its parent

Routing objects are used to partition the space: all the objects in the subtree of O_r (referenced by ptr) are within the distance r ($r > 0$) from O_r. Distance of O_r from its parent is the distance from the object which references the node where the O_r is stored. This information is used to decrease the number of distance computations during the SEARCH operation.

Figure 9 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.

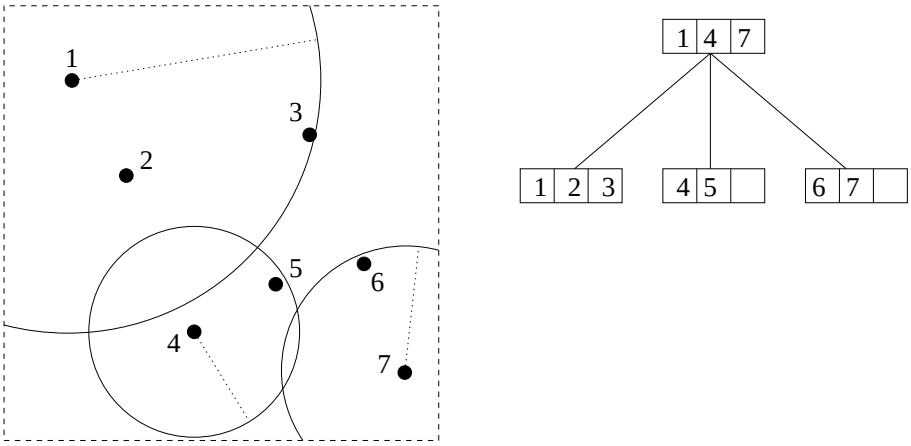

Figure 9: m–tree

5 Data structures in spatial data mining

5.1 Neighborhood graphs

Neighborhood graphs and neighborhood paths

Definition: *Neighborhood graph* G [11, 28] 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. 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 $neighbor$.

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 [n_0, n_1, . . ., n_{k-1}] there

must be edges (n_i, n_{i+1}) for every $0 \leq i < k - 1$. *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 $data$. 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. 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 [11] 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 (see section 2.2) spatial data are stored in structures described in section 1 – points, chains and polygons. Program uses operation `get_Graph` from previous definition (implemented in PostgreSQL [36]) to create neighborhood graph from these data. The graph is stored in database and is used to find neighbors of an object during the knowledge discovery process.

5.2 Amalgamation

In [45] two polygon amalgamation algorithms are described. Operation of polygon amalgamation can be used in spatial data mining where spatial objects (represented by polygons) need to be amalgamated when the user wants to aggregate them. Both these algorithms identify boundary rectangles which are important for the resulting boundary. The first of these algorithms, adjacency method, uses information about objects adjacency to identify boundary rectangles. The adjacency of two objects can be presented with the neighborhood graph. The second algorithm, occupancy method, uses *z–ordering* to partition the space. Z–ordering can be seen as a quad tree with paths described with numbers: The space is divided recursively into four quadrants. Each of these quadrants is identified by a number from 1 to 4. Numbering these quadrants recursively, as they are divided, we get the so called z–value which can be maintained by the one–dimensional access method *B+–tree*. The z–ordering method is modified and each quadrant contains information about its occupancy by amalgamated polygons. Bordering quadrants are not totally occupied.

5.3 Data structures in GWiM

System GWiM was described in section 2.2. It 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 [35] partial solution of this problem was proposed, but it wasn't implemented.

5.4 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).

In [10] the algorithm C4.5 [38], which generates decision trees, is used for image interpretation. This algorithm allows to classify images not only

according to values of the channels of multispectral image, but we can also use some other information such as terrain model or altitude [23].

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. For example, if the classification of the pixel is uncertain, we can check the classification of its neighboring pixels and if e. g. 5 of 8 neighbors are classified as "water", the pixel will be classified as "water", too.

In [26] neighborhood graph is used to interpret topology of objects in the raster image. In this neighborhood graph each pixel is connected to its four neighbors (north, south, west and east) by edge. This graph is stored in database and is used to create neighborhood paths of different lengths. Objects in these paths are used for classification of central pixel (first pixel in the path).

5.5 Other methods

In [32] the application of classification and trend detection in satellite raster images is described. The result of this application is an identification of the trends in the land use changes in the dynamic urban area of Brno city. The source images were preprocessed and then unsupervised classification (based on pixels clusterization) was applied to them. The results of the classification were used to find differences in the land use between images. The methods for finding differences are: subtracting digital values of corresponding pixels, dividing values or comparing the classifications of the pixels.

Another application of the raster images classification is described in [42], where treetops are detected in an airborne image of a forest. The first step of image processing is smoothing out with different filters (both linear and nonlinear). In the next step, treetops are detected. This step is based on the information that treetops have a greater reflectance than their neighborhood. In the final steps, duplicate identifications of tops are filtered.

In [22] the analysis of traffic accessibility with respect to public transport accessibility is described. This analysis requires the information about road infrastructure and time schedules of public transport.

In [29] the distance analysis, network analysis and location–allocation

analysis are used in the analysis of Automatic Teller Machines (ATMs) distribution in Bratislava city. Goal of this application is the information about efficiency of ATMs positions. In this paper the road infrastructure of the city and chosen demographic characteristics are used for the analysis.

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] Beckmann N., Kriegel H. P., Schneider R. and Seeger B.: *The R*-tree: an efficient and robust access method for points and rectangles* Proceedings of the 1990 ACM SIGMOD international conference on Management of data, 1990, Atlantic City, NJ USA, pp. 322-331.

[3] Bentley J.: *Multidimensional Binary Search Trees used for Associative Searching*. Communications of ACM, 18, 9, Sep. 1975, pp. 509-517.

[4] Bigolin N. M., Marsala C.: *Fuzzy Spatial OQL for Fuzzy Knowledge Discovery in Databases.* Proc. of the Second Eurepean Symposium on Principles of Data Mining and Knowledge Discovery (PKDD '98), Nantes, France, Lecture Notes in Computer Science, Springer, 1998.

[5] Bozkaya T., Ozsoyoglu M.: *Indexing Large Metric Spaces for Similarity Search Queries*. ACM Trans. Database Syst. 24, 3 (Sep. 1999), Pages 361 - 404.

[6] Paolo Ciaccia, Marco Patella, Pavel Zezula: *M-tree: An Efficient Access Method for Similarity Search in Metric Spaces.* VLDB 1997: 426-435.

[7] *Data Mining and Knowledge Discovery*: http://www.digimine.com/usama/datamine/

[8] Descartes: http://allanon.gmd.de/and/descartes.html

[9] DBMiner: http://www.dbminer.com

[10] Dobrovolný P., Popelínský L., Kuba P.: *Využitelnost algoritmů strojového učení pro klasifikaci multispektrálního družicového snímku.* Proc. of Conf. GIS... Ostrava 2001.

[11] Ester M., Kriegel H. P., Sander J.: *Spatial Data Mining: A Database Approach.* Proc. of the Fifth Int. Symposium on Large Spatial Databases (SSD '97), Berlin, Germany, Lecture Notes in Computer Science, Springer, 1997.

[12] Fayyad U.M.,Piatetski-Shapiro G., Smyth P., Uthurusamy R. (eds.): Advances in Knowledge Discovery and Data Mining. AAAI/MIT Press 1996.

[13] Finkel R., Bentley J.: *Quad Trees: A Data Structure for Retrieval on Multiple Keys.* Acta Informatica, Vol. 4, No. 1, 1974, pp. 1-9.

[14] Flener P., Popelínský L., Štěpánková O.: *ILP and Automatic Programming: Towards three approaches.* Proc. of 4th Workshop on Inductive Logic Programming ILP'94, Bad Honeff, Germany, 1994.

[15] Gaede V., Günther O.: *Multidimensional Access Methods.* ACM Computing Surveys, Vol. 30, No. 2, June 1998, pp. 170 - 231.

[16] GeoInformatica: http://www.wkap.nl/journalhome.htm/1384-6175

[17] GeoMiner: http://db.cs.sfu.ca/GeoMiner/

[18] GIScience: http://www.giscience.org/

[19] Guttman, A.: *R-trees: a dynamic index structure for spatial searching.* Proc. of SIGMOD Int. Conf. of Management of Data, 1984, pp. 47-54.

[20] Han et al.: *DMQL: A Data Mining Query Language for Relational Databases.* In: ACM-SIGMOD'96 Workshop on Data Mining.

[21] 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).

[22] Horák J.: *Analýzy dopravní dostupnosti a obslužnosti* Proc. of Conf. GIS... Ostrava 2001.

[23] Hyltén A. H., Uggla E.: *Rule–Based Land Cover Classification and Erosion Risk Assessment of the Krkonoše National Park, Czech Republic.* Seminarieuppsatser Nr. 71, Lunds Universitets Naturgeografiska Institution, 2000.

[24] Koperski K., Han J., Adhikary J.: *Mining Knowledge in Geographical Data.* To appear in Comm. of ACM 1998. http://db.cs.sfu.ca/sections/publication/kdd/kdd.html

[25] *ACM International Workshop on Advances in Geographic Information Systems:* http://www.informatik.uni-trier.de/~ley/db/conf/gis/index.html

[26] Kuba P.: *Jazyk pro vyhledávání znalostí v prostorových datech.* Proc. of Conf. DATASEM 2000, Brno, 2000, pp. 213-222, ISBN 80-210-2428-3.

[27] Kuba P.: *Vyhledávání znalostí v prostorových datech.* Master thesis, Faculty of Informatics, Masaryk University, Brno, 2000.

[28] Kuba P., Popelínský L.: *Automatická klasifikace prostorových dat.* Proc. of Conf. GIS... Ostrava 2000.

[29] Kusendová D., Štepitová D.: *Použitie nástrojov GIS v obchodno-služobnej aplikácii.* Proc. of Conf. GIS... Ostrava 2001.

[30] Marsala C.: *Apprentissage inductif en présence de données imprécises : Construction et utilisation d'arbres de décision flous.* Ph.D. thesis, l'UNIVERSITÉ PARIS 6, 1998.

[31] NCGIA: http://www.ncgia.ucsb.edu/

[32] Petrová A., Matuška P.: *Zjišťování změn ve využití země pomocí DPZ (suburbánní oblast Brna).* Proc. of Conf. GIS... Ostrava 2001.

[33] PKDD: http://www.informatik.uni-trier.de/~ley/db/conf/pkdd/index.html

[34] Pokorný J.: *Prostorové datové struktury a jejich použití k indexaci prostorových objektů.* Proc. of Conf. GIS... Ostrava 2000.

[35] Popelínský L.: *Knowledge Discovery in Spatial Data by Means of ILP.* In Zytkow J.M., Quafafaou M.(eds.): Principles of Data Mining and Knowledge Discovery. Proc. of 2^{nd} Eur. Symposium, PKDD'98, Nantes, France. LNCS 1510, Springer Verlag 1998.

[36] PostgreSQL: http://www.postgresql.org/

[37] P-Progol: http://oldwww.comlab.ox.ac.uk/oucl/groups/machlearn/PProgol/ppman_toc.html

[38] Quinlan, J. R.: *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publ. 1992.

[39] Rt4.0: http://www.ncc.up.pt/∼ltorgo/RT/

[40] Sellis T. K., Roussopoulos R., Faloutsos C.: *The R+-Tree: A Dynamic Index for Multi-Dimensional Objects.* Proc. 13th International Conference on Very Large Data Bases, (VLDB'87), Brighton, England, 1987, pp. 507-518, Morgan Kaufmann, ISBN 0-934613-46-3.

[41] Spin!: http://www.ccg.leeds.ac.uk/spin/

[42] Šumbera S.: *Detekce vrcholů korun stromů z leteckých snímků.* Proc. of Conf. GIS... Ostrava 2001.

[43] Tuček J.: *Geografické informační systémy. Principy a praxe.* Computer Press 1998.

[44] Wrobel S., Wettschereck D., Sommer E., Emde W.: *Extensibility in Data Mining Systems.* Proceedings of KDD'96 2nd International Conference on Knowledge Discovery and Data Mining, AAAI Press, 1996, pp. 214–219.

[45] Xiaofang Zhou , David Truffet , Jiawei Han: *Efficient Polygon Amalgamation Methods for Spatial OLAP and Spatial Data Mining.* Proc. of the 6th Int. Symposium on Advances in Spatial Databases (SSD'99), Hong Kong, China, Lecture Notes in Computer Science, vol. 1651, Issue, pp 167-187.

Publications in the FI MU Report Series are in general accessible via WWW and anonymous FTP:

`http://www.fi.muni.cz/informatics/reports/`
`ftp ftp.fi.muni.cz (cd pub/reports)`

Copies may be also obtained by contacting:

**Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic**